Geometry and Texture Recovery of Scenes of Large Scale: Integration of Range and Intensity Sensing.

Ioannis Stamos

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2001

©2001

Ioannis Stamos

All Rights Reserved

Geometry and Texture Recovery of Scenes of Large Scale: Integration of Range and Intensity Sensing.

Ioannis Stamos

This thesis is a systematic approach to the problem of photo-realistic 3–D model acquisition from the combination of range and image sensing. A comprehensive and unique system has been developed in the context of this thesis. This system provides 3-D models of urban scenes with associated 2-D image textures. Our input is a sequence of unregistered range scans of the scene and a sequence of unregistered 2-D photographs of the same scene. Our output is a true texture-mapped geometric CAD model of the scene.

Contents

List of Figures

Acknowledgments

I would like first of all to thank my advisor Peter K. Allen for his constant support over the years in the difficult and in the good times. Without his valuable help, confidence in me and inspiration this thesis would not be possible. I am very grateful to Dr. Michael K. Reed who provided me his solid modeling code and many suggestions. Then I would like to thank professor Shree Nayar for inspiring parts of this thesis and for his valuable moral support over the years. I am grateful to professor John Kender for many important suggestions regarding my research direction. Many thanks to Dr. James Williams and Dr. Fausto Bernardini for being members of my thesis committee and for their interest in my work.

I am grateful to the other members of the Robotics Laboratory: Andrew Miller, Paul Oh, Atanas Georgiev, Paul Blaer and Ethan Gold. We spent a lot of time together, constantly helping each other. I would like to thank Steven Abrams for providing me his sensor planning code and Atanas for his help in scanning many buildings presented in this thesis.

I am grateful to my friends Apostolos Dailianas, Gong Su, Stauroula Sofou, Gulcin Afres, Alexander Konstantinou, Maria Papadopouli, Andreas Prodromidis and Elias Gagas for the many ways they helped in making this thesis possible.

Many thanks to Panagiotis Sebos for his help in scanning the Flat-Iron building.

I would like to thank the "Institution of Ioannis S. Latsis" and Ms. Margarita Kappa. The scholarship and moral support provided is something I will never forget.

Finally my deepest gratitude goes to my parents and my sister Maria. We fought this fight together.

To mother Androniki and father Marinis

Chapter 1

Introduction

The recovery and representation of the 3–D geometric and photometric information of the real world is one of the most challenging and well studied problems in Computer Vision and Robotics research. However, the complexity of the real world objects make it one of the most difficult problems to attack. The visual appearance of real scenes is very hard to interpret due to the interaction of the complex object geometry, the unknown object reflectance properties and the varying scene lighting conditions. This complexity and difficulty is what makes this problem extremely interesting from a research point of view. It also makes this problem almost intractable for a completely automated method in the general case. The importance of this problem, though, is not only its challenge as a research topic. There is a clear need for highly realistic geometric models of the world for applications related to Virtual Reality, Tele-presence, Digital Cinematography, Digital Archaeology, Journalism, and Urban Planning.

Recently, there has been a large interest in reconstructing models of outdoors urban environment including an international workshop held in Japan [Ins,

1999]. The areas of interest include geometric and photorealistic reconstruction of individual buildings or large urban areas using a variety of acquisition methods and interpretation techniques, such as ground-base laser sensing, air-borne laser sensing, ground and air-borne image sensing. The ultimate goal is the reconstruction of detailed models of urban sites (digital cities). Urban Planning is one major application: 1) how does a new building affect the esthetics of local areas of the city? 2) how does the shadow of a new building effect nearby areas? Other applications include micro-land forecasting (prediction of the way a city evolves when new urban regulations are applied), digital city archiving (recording of a city's evolution in time), disaster prevention, visual effects (for entertainment purposes), and journalism. The creation of digital cities drives other areas of research as well: visualization of very large data sets, creation of model data-bases for GIS (Geographical Information Systems) and combination of reconstructed areas with existing digital maps.

Currently, complex photorealistic 3-D models have been constructed for the needs of the entertainment industry by human artists in a pain-staking, extremely slow and error-prone process. Computer games, for example, require high degree of geometric accuracy (in order to allow interaction with the user) and high degree of photorealism (in order to appear visually appealing). That amount of detail in the modeling phase requires a large amount of human interaction.

Recent developments in range-sensing have made possible the acquisition of accurate 3-D scans of outdoor scenes. This technology, if wisely utilized, can be used for automating the modeling phase, since range sensors have already proven their effectiveness in controlled laboratory environments (for instance [Curless and

Levoy, 1996, Reed and Allen, 2000]). Taking these methods out of the controlled laboratory environment and using them at large geometrically complex outdoor scenes poses many difficult challenges. In this thesis, we address these problems to create a novel and unique solution.

1.1 Problem Statement

This thesis deals with the problem of photo-realistic reconstruction of complex 3-D scenes from a set of 3-D range scans and a set of 2-D camera images. The main focus is the development of algorithms and tools for the recovery of geometric and photometric information from urban scenes of large scale. We are dealing with all phases of geometrically correct, photo-realistic 3–D model construction in a systematic way. Our goal is to automate the model building process and place the human out of the modeling loop.

The problem we attack can be described as follows:
Given a set of dense 3-D range scans of a complex real scene from different viewpoints and a set of 2-D photographs of the scene, a) create the 3-D solid model which describes the geometry of the scene, b) recover the positions of the 2-D cameras with respect to the extracted geometric model and c) photorealistically render it by texture-mapping the associated photographs on the model. No a-priori information is used for the relative positions between the range scans and between the 2-D photographs. The only a-priori information that is used is an association of a set of photographs with a particular dense scan. We also assume that our environment is dominated by structures with strong parallelism and orthogonality constraints (e.g. urban scenes).

1.2 Method

The integrated system we developed for the production of photo-realistic geometric models of large and complex scenes is described in figure 1.1. We start with a set of range and brightness images which cover the measured site. The range images are first segmented, and 3-D features of interest are extracted. After all range images are expressed in the same coordinate system a volumetric solid geometric model which expresses the geometry of the scene is computed. Finally, the relative positions of the brightness cameras with respect to the 3-D model are automatically recovered and the photographs are mapped on the geometric model in order to provide a photorealistic view of the scene.

1.3 Comparison with other methods

There are numerous approaches for the photo-realistic reconstruction of a general 3–D scene. The goal of all methods is the realistic rendering of the captured scene. There are two major approaches in this direction: purely geometric and image-based rendering methods. In the first category, a complete geometric model of the scene is acquired. Renderings from novel viewpoints can be computed by mapping real photographs of the scene on the geometric model (texture-mapping).

Image-based rendering methods on the other hand attempt to produce photorealistic renderings by by-passing the hard geometric modeling step. In image-based rendering methods a set of photographs of the scene generate a light field that is being interpolated in order to generate novel renderings from different viewpoints. The inherent drawbacks of those methods are a) the inability to cover

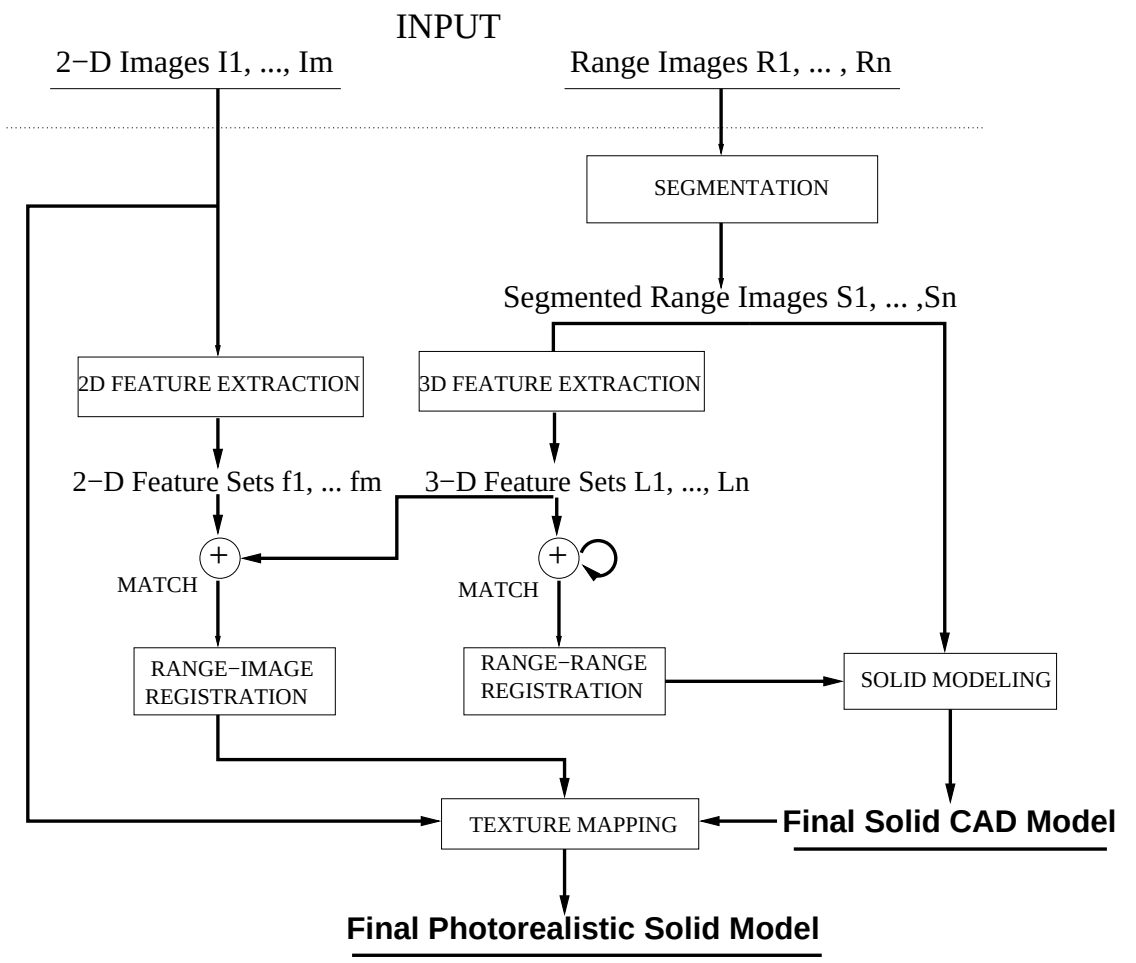

Figure 1.1: System for building geometric and photometric correct solid models.

every possible novel viewpoint, b) the fact that a large number of images is needed for undistorted renderings and c) the inability to model novel illumination conditions. In order to overcome these limitations image-based rendering methods use a simplified geometric model of the scene.

Below we classify geometry-based systems whose goal is the photorealistic recovery of scenes according to the amount of geometry information that is being recovered (a detailed overview of related work is presented in **chapter 2**).

Sparse and Irregular Geometry/Brightness Camera Classical computer vision methods based on stereo or structure from motion provide results which cover parts of the scene (those parts where stereo measurements are reliable) in an irregular manner. Major limitations incluse the fragility with respect to occlusion and lighting variations.

Simplified Geometry/Brightness Camera Human interaction results in simplified geometric representations intended to be utilized by image-based rendering techniques (i.e. [Debevec *et al.*, 1996]). Also, simplified geometry is recovered when a-priori constraints are implied in the reconstruction process (i.e. [Teller, 2000]).

Dense and Regular Geometry/Range Sensor Dense and regular geometric descriptions are only possible when the acquisition involves the utilization of range sensing technology. Major representatives of comprehensive systems which combine range and image sensing in large environments include the following [VIT, 2000, Yu, 2000, Sequiera *et al.*, 1999, Zhao and Shibasaki, 1999]. Systems which provide excellent results in the construction of solid models of small objects are presented in [Curless and Levoy, 1996, Reed and Allen, 2000, Bernardini *et al.*, 1999].

This thesis is placed in the last category **Dense and Regular Geometry**. It shares the system-oriented approach of [VIT, 2000, Sequiera *et al.*, 1999, Zhao and Shibasaki, 1999], the use of a-priori constraints wrt scene structure (but in a much more limited sense) of [Teller, 2000] and the construction of solid 3-D models of [Curless and Levoy, 1996, Reed and Allen, 2000]. The contribution

of this thesis is not only the development of a new system which combines the aforementioned properties, but also the development of algorithms in the areas of range segmentation and range to image registration, the extension of a solid modeler to handle large scale scenes and the provision of a framework for incorporation of range sensor planning.

1.4 Contribution

We see the contributions of this thesis in the following areas:

1. Creation of a comprehensive system for the automatic generation of geometrically correct and photometrically accurate 3-D models.

2. Development of efficient algorithms for the segmentation and feature extraction from 3-D range scans [**chapter 3**].

3. Solid modeling in large-scale environments [**chapter 4**].

4. Development of algorithms for automated registration between range and image data-sets [**chapter 5**].

5. Providing an initial framework for incorporation of sensor planning utilities and development of an interactive tool for sensor planning [**chapter 6**].

Chapter 2

Related Work

The reconstruction of geometric 3–D models of the world is one of the most important goals of Computer Vision research. Those 3–D models, enhanced with photometric observations provide representations that are highly desirable by graphics applications. In this chapter, we will cover related work in the areas needed to build a complete photorealistic-model creation system. In particular, we will present work in the areas of range segmentation and feature extraction, range to range and range to image registration (automated pose estimation). Finally, we will describe unique and representative systems whose goals are similar to ours and conclude with methods to visualize registered range and image data-sets.

2.1 Range Segmentation and Feature Extraction

In this section we present an overview of related work in the area of range segmentation and feature extraction. The input is a range scan of a real 3-D scene, that is the input is a cloud of 3-D points which cover the scene from a particular viewpoint.

The output of the segmentation is a set of disjoint surfaces which approximate the measured 3-D cloud.

The work of [Besl and Jain, 1988] is an important general range-image segmentation algorithm[1] which works in two steps. In the first step a coarse segmentation is achieved by assigning one of eight possible local surface types to every range-point according to the sign of the corresponding local Gaussian curvatures. This coarse segmentation provides connected regions of coarsely classified points. The interiors of the coarsely segmented regions are used for the initial fit of one of four bivariate surfaces (planar, biquadratic, bicubic or biquartic) providing seed regions (with associated seed surfaces). In the second step the seed regions (and their associated bivariate surfaces) grow by including neighboring points whenever the fit to the associated surface is preserved. The complications of this algorithm involve the big number of thresholds that need to be specified and the instability that can be caused due to the incorporation of Gaussian curvatures (which involve second degree derivatives). The lack of precision on the estimation of Gaussian curvatures is analyzed in [Trucco and Fisher, 1995], where it is suggested that Gaussian curvatures should not be used for planar segmentation routines. Segmentation of non-regular but sparse and noisy data-sets is presented in [Guy and Medioni, 1997]. [Boyer *et al.*, 1994] presents a segmentation algorithm similar to [Besl and Jain, 1988]. Another work related to segmentation but in a different domain (classification of a set of linear segments extracted from stereo into groups of planar regions) is described in [Zhang and Faugeras, 1994].

In [Hoffman and Jain, 1987] a segmentation algorithm, based on clustering

[1]This algorithm can be applied to intensity images as well.

the 6-dimensional space of range points with their associated normals, is presented. After the initial clustering surfaces of similar orientation are merged into larger groups. In the approach followed by [Jiang and Bunke, 1994] the range-scan lines are divided into linear segments and the segmentation proceeds the grouping of those segments into planes. [Koch, 1996] on the other hand provides a simplistic solution to planar segmentation by histogramming the surface normals and extracting clusters of similarly oriented 3-D points. A comparison of a number of different range-segmentation algorithms can be found in [Hoover *et al.*, 1996]. The above list is not exhaustive since range segmentation was (and still is) a very active area in Computer Vision Research. Complementary approaches can be found in [LaValle and Hutchinson, 1995, Leonardis *et al.*, 1995, Newman *et al.*, 1993, Sabata *et al.*, 1993, Taylor *et al.*, 1989].

Finally [Yu, 2000] developed a range segmentation algorithm which is based on the normalized cut framework for image segmentation [Shi and Malik, 1997]. Yu segments a set of overlapping range scans by utilizing the average position, normal and laser intensity of connected clusters of range points (the algorithm operates on connected clusters of range points and not on the range points themselves for efficiency reasons). The algorithm results in scene over-segmentation which needs to be corrected manually (by merging connected regions).

The segmented surfaces extracted by algorithms like the one described previously transform the 3–D point cloud into a set of 2–D surfaces which cover the cloud. Those surfaces can also be used for the identification of 1–D curves at their boundaries or at their intersection with neighboring surfaces. However, the previously described segmentation algorithms do not proceed in that direction. Related

work in 3-D edge detection can be found in [Zhang, 1993, Jiang and Bunke, 1999, Monga *et al.*, 1991, Parvin and Medioni, 1988].

2.2 Pose Estimation (3–D and 2–D)

Pose estimation and camera calibration are two of the most important problems in Computer Vision and Robotics. **Pose estimation or External camera calibration** is the problem of estimating the location (position and orientation) of an internally calibrated camera with respect to a 3–D model. **Camera calibration** is the problem of estimating the internal camera calibration parameters of a camera and its position with respect to a 3–D model. The internal camera parameters include the principal point, the effective focal length and the distortion coefficients. The solution to both problems requires the knowledge of a set of corresponding 3–D and 2–D features, that is 3–D features (3–D points or 3–D lines) and their 2–D projections (2–D points or 2–D lines) on the image plane (those features are usually called *landmarks* in the literature). A minimum number of points (or lines) is needed for the existence of a robust solution. In both problems two-dimensional line features are advantageous because they can be reliably extracted and are prominent in man made scenes.

When point landmarks are being used for estimating pose then *three point* correspondences generate four possible solutions, *four coplanar* correspondences generate a unique solution, while *four* points in general position result in two possible solutions. Six or more points always produce a unique solution (for more details see [Horaud *et al.*, 1989]). There are many approaches for the solution of pose estimation problem from point correspondences [Fischler and Bolles, 1981,

Liu *et al.*, 1990, Dhome *et al.*, 1989, Oberkampf *et al.*, 1996, DeMenthon and Davis, 1995, Quan and Lan, 1999]. In the case of line landmarks three lines in general position are enough [Liu *et al.*, 1990, Kumar and Hanson, 1994, Horaud *et al.*, 1997, Christy and Horaud, 1999]. However, due to noisy measurements the minimum number of correspondences is not enough for a robust solution. Also, the accuracy of the solution depends heavily on the spatial configuration of the landmarks.

In Liu's [Liu *et al.*, 1990] work, a minimum number of eight line or six point correspondences are needed in order to linearly solve for camera rotation. A non-linear solution is possible if at least three line or two point correspondences are given. The translation can be solved with a linear method (assuming a known rotation) if three line or two point correspondences are known. Non-linear methods are considered more robust with respect to image noise. However those methods fail in the case of gross errors in the correspondence between image features (least-squares approaches fail in the case of outliers). That problem is attacked by Kumar and Hanson [Kumar and Hanson, 1994], where a method for pose estimation in the presence of outliers and a sensitivity analysis is presented.

Finally, Fischler and Bolles [Fischler and Bolles, 1981] provide a robust solution in the presence of gross correspondence errors between 3-D points and 2-D image points. In this case three pairs of corresponding 3-D and 2-D features are randomly selected and the pose of the camera is computed. If most of the other pairs of corresponding 3-D and 2-D features are in agreement with the computed pose a least-squares algorithm provides the final pose estimate. Otherwise a new set of three corresponding 3-D and 2-D features is selected and a new estimate is computed.

Those problems are of fundamental importance in computer vision and robotics research since their solution is required or coupled with stereo matching, structure from motion, robot localization, object tracking and object recognition algorithms.

2.3 Automated pose estimation algorithms

This section presents related work in the field of automated pose estimation. The problem can be defined as follows: given a 3-D model of a scene and a brightness image (taken by a 2-D camera) of the same scene compute the relative position of the camera with respect to the 3-D model. The solution to this problem involves the extraction of features of interest from the 3-D and 2-D data sets and the automated matching between a set of 3-D and 2-D features.

There are two classes of methods for geometric feature matching [Cass, 1997]: a) *Correspondence space methods*, where a search for consistent feature correspondences is performed in the feature space and b) *Transformation space methods* where the search is performed in the transformation space in order to geometrically align the model with the image.

One major representative of the correspondence space methods is The Alignment method which is based on the following four basic steps (a hypothesis-verification scheme):

1. Select a hypothesized set of model and matching image features.

2. Use the matched features to compute a transformation that brings the model into the image coordinate system and then project the model into the image.

3. Decide what region in the image to search about each projected model feature to find candidate image features and,

4. Use the candidate image features to accept or reject the hypothesis.

In the early work of [Huttenlocher and Ullman, 1990] an automated pose-estimation in the context of the Alignment is presented. A brute-force matching algorithm which checks all possible matches between 3-D and 2-D point features is based on an affine approximation of the perspective projection (which makes the transform computation a linear algorithm). A minimum of three model and image point correspondences is needed. The major drawback is the large number of hypotheses that need to be tested. In the alignment framework [Alter and Grimson, 1997] present results in the verification step (step 4 of the Alignment method). In [Gandhi and Camps, 1994] a robust solution to the problem of the selection of the next point to match after an initial correspondence has been achieved is presented and analyzed under the assumption of noisy measurements.

In [Cass, 1997] a novel formulation of the problem is presented. This formulation leads to a method which uses both the correspondence and transformation space concepts. The presented algorithm (which involves the computation of geometric arrangements) is polynomial-time on the number of extracted features and an affine approximation to the perspective projection is used. The work of [Wells, 1997] transforms the pose estimation problem to a well defined optimization problem following an *align, refine* and *verify* framework. [Jacobs, 1997] attacks the model recognition problem by analytically characterizing all 2-D images (set of 2-D points) that a group of 3-D features may produce under all possible poses under an affine transformation model. In [Hausler and Ritter, 1999] all possible triplets (they

provide four possible solutions for the desired transformation) or quadruples (they provide a unique solution) of matched 3-D and 2-D features are used in order to update a six-dimensional Hough table. The hough table encodes a quantization of the pose space and a peak in that space corresponds to the computed final pose. Finally [Jurie, 1999] present an approach which refines an initial coarse pose estimate (which defines a box in pose space) by minimizing a quadratic objective function which encodes the conditional probability of a match between features under the assumption of a particular pose.

Automated pose estimation is not part of most systems which acquire complex environments via means of range sensing. Most systems which recreate photo-realistic models of the environment by a combination of range and image sensing [VIT, 2000, Pulli *et al.*, 1998, Zhao and Shibasaki, 1999, Sequiera *et al.*, 1999] solve the range to image registration problem by fixing the relative position and orientation of the camera with respect to the range sensor (that is the two sensors are rigidly attached on the same platform). The camera calibration is done off-line by sensing scenes of known geometry. Recently Yizhou Yu [Yu, 2000] developed an automatic method for the image to range registration problem, where a static arrangement of a range and camera system is not assumed. His solution involves the placement of artificial 3–D markers (spheres of known size and reflectance) in the scene which are detected in the range and image acquisition phase. That leads to the undesirable feature of acquiring an artificially altered scene model. Also, in [McAllister *et al.*, 1999] an optimization of the rotational parameters of the image to range transformation is provided (in the described system the range and image sensor share the same center of projection by manually replacing part of the range

sensor with an internally calibrated digital camera). Finally in [Neugebauer and Klein, 1999] an automatic image to range registration method is provided after an initial solution to the problem is achieved through interactive selection of matching 3-D and 2-D points.

2.4 Range-range registration (3–D / 3–D)

Registering two range images when two sets of corresponding 3–D points have been identified can be done using a quaternion-based non-linear optimization method as described in [Horn, 1987]. The automation of the process of establishing point correspondences was presented first in [Besl and Mckay, 1992]. This is the widely used Iterative Closest Point algorithm, where the rigid transformation between two views is iteratively refined, while larger sets of corresponding points between views can be extracted after each refinement step. This algorithm has been extended to work for the alignment of general meshes (in the original algorithm of Besl and McKay one of the meshes must be a proper subset of the other) in [Turk and Levoy, 1994]. Here the alignment is part of a larger system where partial meshes, describing distinct views of an object, are being merged into one non-redundant mesh. Both methods require the meshes to be spatially close with respect to each other in order for an initial set of closest point correspondence to be established. A different approach is followed in [Johnson and Hebert, 1997], where the latter constraint of spatial closeness between meshes is removed. Here the initial list of corresponding points is extracted by using a pose-invariant representation for the range images. Each oriented-point in the range image is represented by a 2–D image, called spin-image. The spin-image of an oriented point encodes the spatial relationship of all

range-image points with respect to the particular oriented point. An initial list of correspondences is extracted by matching individual spin-images between range images. This list is later refined by means of the Iterative Closest Point algorithm. The spin-images algorithm is part of a larger project [ARTISAN, 2000] for semi-automatic reconstruction of industrial environments from range images. Earlier approaches in registration by matching pose-invariant 3-D representations include the following: [Bergevin *et al.*, 1995, Chua and Jarvis, 1996, Gueziec and Ayache, 1994, Ikeuchi *et al.*, 1996].

Integration of both range and intensity information in the 3-D registration process is described in [Weik, 1997, Johnson and Kang, 1997, Magee *et al.*, 1985, Koch, 1993]. In Weik's approach [Weik, 1997] the intensity information is used for the selection of the set of corresponding points. Lucchese [Lucchese *et al.*, 1997] proposes a method based on 3–D Fourier transform for the registration of 3–D solids (the method does not rely on specific 3–D features but on the frequency information contained in the 3–D shape). In this case the intensity information is used only for the disambiguation of the shape-based registration results (results are presented for synthetic data-sets only).

2.5 Complete Systems

The topic of this section is the description of a number of representative systems whose goal is the photorealistic reconstruction of real scenes by the utilization of 2-D imagery or combination of range and 2-D imagery.

2.5.1 A-priori 3-D constraints: simplified geometry

In Shum's [Shum *et al.*, 1998] and Becker's [Becker and Bove, 1995, Becker, 1997] work, the user selects image lines which are grouped according to their 3–D scene relationships (parallel or orthogonal with respect to each other). This grouping is done by the user. A rough polyhedral model of the scene is created and the pose of each camera is computed with respect to this model. Becker puts the emphasis in internal camera calibration (focal length, principal point and distortion parameters) using image vanishing points (computed from image lines selected by the user). The most successful approach in this direction is the Façade system [Debevec *et al.*, 1996]. A parameterized description of the scene provided by the user is optimized wrt to camera positions and line correspondences (which are again provided by the user).

Human interaction leads to lack of scalability wrt the number of processed images of the scene and to the computation of simplified geometric descriptions of the scene. In the case of Façade the CAD primitives are simple polyhedral solids whereas in the case of Becker the scene is assumed to be polyhedral. That means that if one wants to describe all the details of a complex building the amount of a–priori human information would be enormous. However those models are acceptable for rendering purposes because texture-mapping hides the lack of captured details.

We are using Façade's idea of model to image line correspondence for estimating camera pose and Becker's idea of exploitation of parallel and orthogonality constraints for camera self–calibration. In our case though the model building phase is automatic using 3–D range measurements. *Thus we aim to minimize the amount of user interaction.*

2.5.2 Using both range and image information

The VIT group [VIT, 2000, Beraldin *et al.*, 1997, El-Hakim *et al.*, 1997] has built a mobile platform which carries a range and several camera sensors and acquire geometric and photometric information of indoor and outdoor scenes. This method is the closest to our method since it combines range with image sensing.

The relative position of the range sensor and the nine cameras is always fixed and known from a calibration procedure which is performed off–line. The information provided by all sensors is used in order to automatically register successive viewpoints by means of a global optimization procedure. The range information is used for the building of the solid model, whereas texture information is provided by an additional high resolution camera which lies on the mobile platform.

The problems of this approach are:

1. The need for accurate and stable internal calibration of the sensors. The sensors do not self-calibrate or adjust while measurements are taken. This is a limitation since off–line calibration is required regularly.

2. The need of user interaction for the registration of the high resolution camera with the range sensor (finding corresponding points).

Another system which wisely combines range (multi-view stereo) with image sensing is the one used in the Pietá Project [Bernardini and Rushmeier, 2000]. The system recovered in great geometric and photometric detail the 3-D model of the Florentine Pietá statue of Michelangelo. The used sensor provided register range and image data and a photometric stereo technique was used for the recovery of the albedo maps of the object's surfaces. Hundreds of scans were registered and

the final non-redundant mesh was computed [Bernardini *et al.*, 1999].

In [Zhao and Shibasaki, 1999] a set-up which involves a laser scanner and a camera sensor at a fixed position wrt scanner is used in order to provide panoramic textured range images. The problem of range registration between a large sequence of range images is solved based on the assumption that neighboring range images are horizontally aligned (that is the viewing direction is parallel to the ground plane). In [Sequiera *et al.*, 1999] another approach to the problem is being presented.

2.5.3 Using only image information

Over-constraining the problem

Teller's approach [Teller, 2000] addresses the limitations of the methods described in section 2.5.1, that is lack of scalability wrt the number of input images used and simplicity of the acquired model. Teller's group acquires and processes a large amount of pose–annotated[2] high-resolution spherical imagery of the scene. The transformation (rotation and translation) between nearby mosaics is computed through the match of vanishing points (rotation) [Antone and Teller, 2000b] and through a Hough transform and expectation maximization technique (translation) [Antone and Teller, 2000a]. The extracted geometry consists of vertical facades with associated textures which are the result of correlation of a number of different images [Coorg and Teller, 1999]. A relief estimation in order to increase the complexity of the vertical facades along with a consensus estimation for the texture facades is presented in [Taillandier, 2000]. The whole project is very promising and it is attacking the problem of model generation of a large urban scene in a brute-force

[2]GPS measurements provide a coarse pose estimate for every spherical mosaic.

yet unique manner. The major problem so-far is the simple acquired geometry (vertical facades). The relief estimation provides hope, however methods based on images alone are not able to capture highly detailed architectural environments.

From Images to Graphical Models: Irregular Geometry

Zisserman's group in Oxford [Fitzgibbon and Zisserman, 1998] works towards the fully automatic construction of Graphical Models of scenes when the input is a sequence of closely spaced 2–D images (video sequence). Their system couples the matching of 2–D point features in triples of consecutive images with the computation of the fundamental matrices between pairs of images and trifocal tensors between triples of images (projective reconstruction). The estimation of the fundamental matrices is based on the RANSAC robust estimator of Fischler and Bolles [Fischler and Bolles, 1981] which was discussed previously (section 2.2) in this document. The projective reconstruction and camera pose estimation is upgraded to an Euclidean one by means of auto-calibration techniques [Pollefeys *et al.*, 1998]. Finally, the registration between successive coordinate frames of image triples is based on the Iterative Closest Points (ICP) algorithm [Besl and Mckay, 1992]. The result of the above procedure is a set of 3–D points which irregularly sample the real 3–D geometry of the scene. The RANSAC estimator is used again for the fitting of planar faces on the 3–D points and a VRML Graphical model is constructed.

This work shows how far purely image–based methods have gone but also points out the following inherent limitations:

1. Sparse depth estimates which depend on the texture and geometric structure of the scene.

2. The resulting CAD model is a crude approximation in the areas which do not support 3–D measurements.

In more recent work of the same group, a–priori knowledge regarding the scene is utilized in the 3-D extraction phase [Baillard and Zisserman, 2000, Montiel and Zisserman, 2001].

2.6 Rendering registered range and intensity data

Registered range and intensity information is what is expected in most cases in image–based rendering graphic systems. Chen and Williams [Chen and Williams, 1993] is the first paper to discuss efficient rendering of registered range and image data. Others [McMillan and Bishop, 1995, Rademacher and Bishop, 1998, Shade *et al.*, 1998] attack the problem of efficient rendering of scenes of large scale when the input is a set of registered range and intensity images.

Coorg in his PhD thesis [Coorg, 1998] (relevant technical report [Coorg and Teller, 1998]) provides an initial solution to the problem of combining textures from different viewpoints in outdoor environments. His approach attacks the problem of texture occlusion from objects which are not modeled in the 3–D model (such as trees, vegetation or occluding buildings). The solution is based on a median-extraction technique under the assumption that the "correct" texture is visible from most images.

Pulli in [Pulli *et al.*, 1997] and Debevec in [Debevec *et al.*, 1996, Debevec *et al.*, 1998] provide efficient solutions for view–dependent texturing, without handling the problem of texture occlusion. That means that parts of the scene which do not

correspond to the modeled object but appear on the input images are erroneously texture-mapped on the model. Both methods blend information provided by images registered with the 3-D model of the scene in the rendering phase. During rendering at each virtual viewpoint the images of the closest original viewpoints are selected. Those images are texture–mapped on the 3-D model and are rendered from the virtual viewpoint. The blending is done by means of weighted averaging of the original images as they are texture–mapped on the 3–D model. The weights express the orientational deviation of the selected images from the virtual viewpoints. Pulli on the other hand, uses a directional weight as well as sampling quality and feathering weights.

Finally Becker in his PhD thesis [Becker, 1997] describes a method to statically texture–map a 3–D model. In this case the color assigned to the polygonal surface elements of the 3–D model does not depend on the position of the virtual camera (unlike [Debevec *et al.*, 1998, Pulli *et al.*, 1997]). Becker blends color information provided by different original views of the scene taking into account the different spatial resolutions of the real images.

All methods have to handle the problem of visible–surface determination in order to texture map original images only on visible polygons. In Pulli's approach this is done in image space using graphics hardware whereas Debevec proposes a hybrid approach where both image space (graphics hardware) and object space (occlusions between 3–D polygons) methods are used.

Chapter 3

Range Imaging, Segmentation & 3–D Feature Detection

3.1 Introduction

We begin our discussion of our system by describing range imaging. The individual range-images which the range-sensor provides are the result of dense sampling of visible surfaces in large urban scenes. Typically we get 1K by 1K range samples (that means 1 million range samples) with a spatial resolution of a few centimeters. The sampling of the scene is regular. Thus, smoothly varying parts of the scene (e.g. planar or cylindrical surfaces) are sampled with the same rate as non-smooth surfaces (e.g. parts of the scene with orientation discontinuities). If we are able to identify those smoothly varying parts then we can represent them with a fewer number of parameters.

In this chapter, we present some important characteristics of range-sensing technology. Then we formulate the range segmentation and 3–D line detection

problems. We discuss the algorithms we have developed [Stamos and Allen, 2000a, Stamos and Allen, 2000b]. Finally, we conclude with a presentation of our results. Figure 3.1 presents the range-segmentation and 3–D feature extraction modules as part of the whole system.

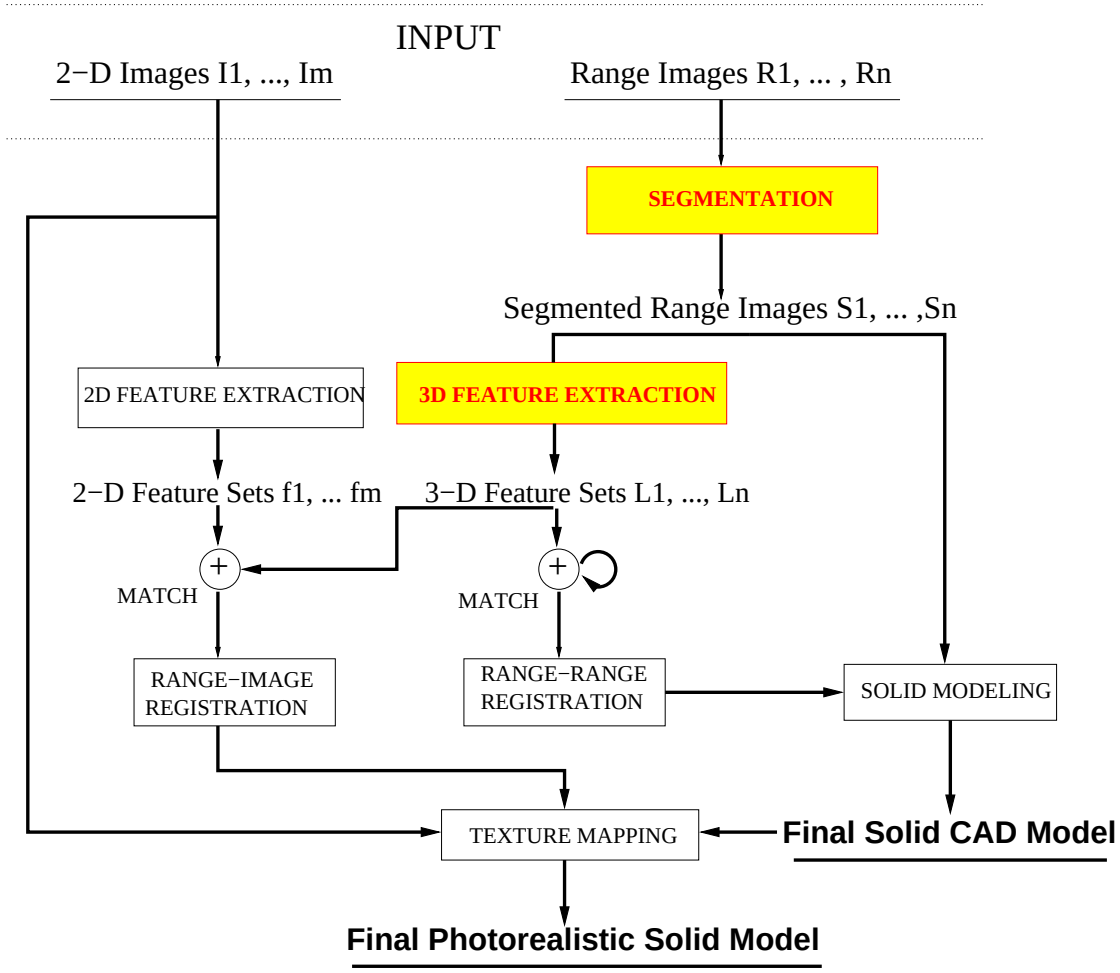

Figure 3.1: Range segmentation and 3–D feature extraction as part of our system.

3.2 Range-Sensing Technology

Laser-range sensing provides *a dense and regular sampling* of the 3–D scene. It overcomes the major limitations of passive techniques by not relying on the visual appearance of 3–D objects on 2–D images. Instead it samples directly the 3–D surfaces by emitting regular patterns of energy beams into the scene and measuring the returned response. The laser is actually "touching" the 3–D surfaces with a controlled source of energy.

Laser-range sensing systems measure the distance between the range-sensor and the 3–D scene along a 3–D ray by emitting a laser-beam towards the scene. Then the range-sensor measures spatial or time properties of the reflected beam and calculates the depth of the 3–D point in the beam direction. The beam can be rotated by a set of two mirrors and cover the volume of a 3–D prism between the sensor and the 3–D scene. Extensive reviews of early range-sensing technology can be found in [Besl, 1988, Poussart and Laurendeau, 1989].

Laser-range sensing techniques are divided into two major categories based on the way they interpret the reflected laser beam:

Triangulation Sensors Sensors in this category measure the one-dimensional disparity between the emitted laser beam and the reflected beam on a laser-sensitive CCD array. The basic principle is again triangulation.

Time-of-Flight Sensors Sensors in this category measure the time that the laser-beam takes to reach the object surface and then to return back to the sensor. Since, the velocity of the laser beam is known (it is the speed of light) the distance between the sensor and the object surface is calculated. The laser-

range sensor [Cyrax Technologies, 2000] we used belongs to the latter category (time-of-flight).

Range image representation

We mathematically represent a range image with a set $\{r(i,j), i = 1 \ldots N, j = 1 \ldots M\}$ of 3-D points $r(i,j)$. The indices i, j define the position and orientation of the laser-beam which produces the 3-D point $r(i,j)$. The arrangement of the laser-beams which sample the scene is shown in figure 3.2a. All rays which belong to the same column or the same row are coplanar. The angle β between successive column-planes is constant and the angle between successive row-planes α is also constant. It is possible that $r(i,j) = \infty$ either due to the fact there is not any opaque object along the laser-beam direction (i,j)[1] or due to the sensor's inability to measure a point in this direction.

Thus the laser-beams are indexed on a 2–D rectangular grid. On this grid it is possible to define connectivity relationships. Figure 3.2b displays a 8-neighborhood around a grid point. The notion of connectivity in the 2–D grid can be transformed to connectivity in the domain of 3–D points. In the rest of the chapter, when we talk about connected regions we refer to connected regions on the 2–D grid. Those connected regions have corresponding regions in the domain of 3–D points.

[1]That means that there is no object up to a maximum distance M_{dist} from the sensor. This distance is a property of the laser sensing technology. In our case $M_{dist} = 100$ meters.

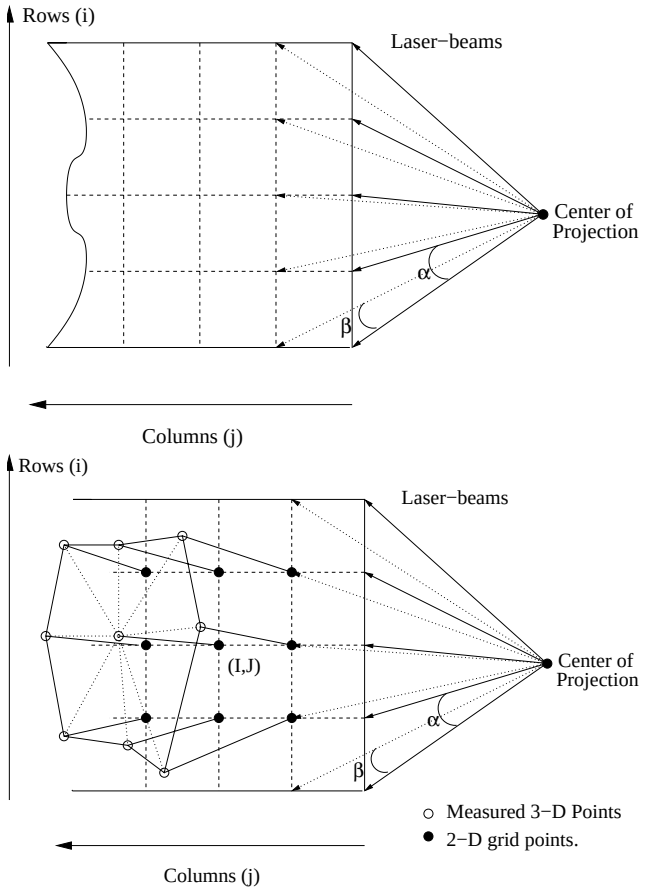

Figure 3.2: **a) Top.** Arrangement of laser-beams for a range sensor with a single center of projection. The laser-beams are indexed on a 2-D rectangular grid. **b) Bottom.** 8-neighborhood around the 2-D grid point (I, J). The grid connectivity is transformed to connectivity in the domain of 3–D points.

3.3 Segmentation

We follow the formulation introduced by [Besl and Jain, 1988]. Our goal is to segment the range image $\{r(i,j), i = 1 \ldots N, j = 1 \ldots M\}$ into a set of clusters $\{C_{null}, C_1, \ldots, C_n\}$. Each cluster $C_i, i \geq 1$ is defined over a connected domain $\{r(i,j)\}$ of 3–D points and it corresponds to a smoothly varying surface S_i of the object. Also no two clusters overlap, that is $C_i \bigcap C_j = \emptyset, \forall i,j : i \neq j$. Finally the number of points that support each cluster is larger than a user defined threshold

T_{size}. The special symbol C_{null} corresponds to the cluster of 3–D points which cannot be classified to any surface. Every 3–D point belongs to one and only one cluster.

Each cluster $C_i, i \geq 1$ is represented by a set of parameters $\mathcal{P}(S_i)$ which define the surface S_i of infinite extent where the points of the cluster lie, and by the sequence of those range points $\{r(i,j)\} \subset C_i$ which define the outer and inner boundaries of the cluster (see figure 3.3). A range-point belongs to the boundary of the cluster if at least one of its 8-neighbors in the 2–D grid (figure 3.2) is not a member of the cluster. The outer boundary $\mathcal{O}(S_i)$ is the one that encloses all range-points of the cluster, whereas the inner boundaries $\mathcal{I}_k(S_i)$ are holes inside the cluster. So, a cluster is represented as follows:

$$C_i = (\mathcal{P}(S_i)|\mathcal{O}(S_i), \mathcal{I}_0(S_i), \ldots \mathcal{I}_K(S_i)).$$

Our final goal is to extract 3–D curves of finite extent at the intersections of adjacent surfaces S_i. That is our goal is to generate a list of curves

$$\mathbf{L} = \{\mathcal{L}(S_{i0}, S_{j0}), \mathcal{L}(S_{i1}, S_{j1}), \ldots, \mathcal{L}(S_{iW}, S_{jW})\}$$

The symbol $\mathcal{L}(S_{iK}, S_{jK})$ corresponds to the curve of finite extent which is the intersection of the surfaces S_{iK} and S_{jK}. Those surfaces have defined boundaries according to the formulation described above.

3.4 Algorithm

The outline of our segmentation algorithm is the following (figure 3.4):

Point Classification A local plane is being fit in the $k \times k$ neighborhood of every 3–D point. If the fit is acceptable the point is classified as **locally planar**

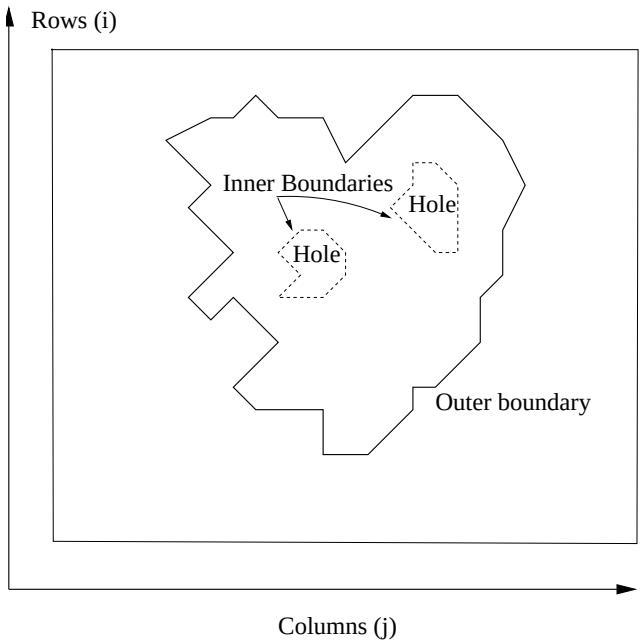

Figure 3.3: Cluster boundaries defined as sequence of points on the rectangular grid over which the range image is defined.

otherwise is classified as **non–planar**. Finally if the number of sensed points in the $k \times k$ neighborhood is not enough to produce a reliable fit the point is classified as **isolated**.

Cluster Initialization Create one cluster for every *locally planar* point.

Cluster Merging Merge clusters from the initial cluster list.

Surface Fit Fit a plane to the points of each cluster.

Boundary Extraction Extract the boundaries of each cluster.

The following sections describe every module of the algorithm.

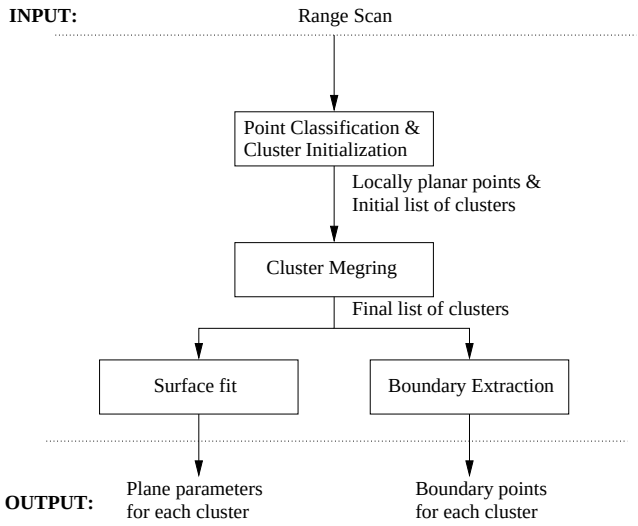

Figure 3.4: Range segmentation algorithm.

3.4.1 Point Classification and Cluster Initialization

In the **Point Classification** phase a plane is fit to the points $\mathbf{v_i}$ which lie on the $k \times k$ neighborhood of every point P. The normal $\mathbf{n_p}$ of the computed plane corresponds to the smallest eigenvector of the 3 by 3 matrix $A = \Sigma_{i=1}^{N}((\mathbf{v_i} - \mathbf{m})^{\mathbf{T}} \cdot (\mathbf{v_i} - \mathbf{m}))$ where $\mathbf{m}$ is the centroid of the set of vertices $\mathbf{v_i}$. The smallest eigenvalue of the matrix A expresses the deviation of the points $\mathbf{v_i}$ from the fitted plane, that is it is a measure of the quality of the fit. If the deviation is below a user specified threshold P_{thresh} the center of the neighborhood is classified as *locally planar* point.

The result of the above step is a list of initial clusters $\mathbf{IC} = \{IC_1, IC_2, \ldots, IC_n\}$. Every cluster IC_i contains one **locally planar** point. We also have the cluster C_{null} which contains all **non–planar** and **isolated** points. The points of this cluster are not considered in the later steps of the algorithm and they can not be parts of any segmented surface.

3.4.2 Cluster Merging

The next step is to merge clusters of the initial list **IC** in order to produce a final list **C** which contains a minimum number of clusters of maximum size. Each cluster in the final list is defined as a set of 3–D points which are connected and which lie on the same surface. Formally, the points of every final cluster represent segmented surfaces that can be approximated by small local patches of similar position and orientation at the point-level. That is, the local planes of every pair of neighboring points are very close with respect to each other (see figure 3.5). In order to generate clusters which have the above attribute we need to define a metric of planar similarity of two neighboring 3–D points.

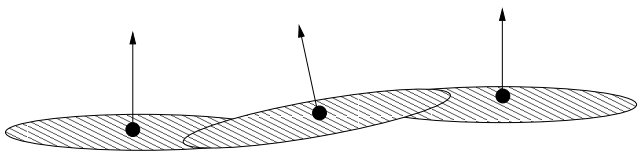

Figure 3.5: Neighborhood of points inside a cluster. The local planes fit around each point are very close wrt each other.

We introduce a metric of co–normality and co–planarity of two planar patches. This metric is very important because it drives the merging of neighboring clusters. Two adjacent *locally planar* points are considered to lie on the same planar surface if their corresponding local planar patches have similar orientation and are close in 3D space. Figure 3.6 displays two local planar patches which have been fit around the points P_1 and P_2 (Point Classification). The normal of the patches are $\mathbf{n_1}$ and $\mathbf{n_2}$ respectively. The points P_i' are the projections of the points P_i on the patches. The two planar patches are considered to be part of the same planar surface if the following two conditions are met:

Co–normality measure The first condition claims that the patches should have identical orientation (within a tolerance region), that is the angle $\alpha = \cos^{-1}(\mathbf{n_1} \cdot \mathbf{n_2})$ is smaller than a threshold α_{thresh}.

Co–planarity measure The second condition is that the patches lie on the same infinite plane. The distance between the two patches is defined as $d = \max(|\mathbf{r_{12}} \cdot \mathbf{n_1}|, |\mathbf{r_{12}} \cdot \mathbf{n_2}|)$, where $\mathbf{r_{12}}$ is the vector connecting the projections of P_1 and P_2 on their corresponding local planes (see figure 3.6). This distance should be smaller than a threshold d_{thresh}

Thus, we defined the predicate $CoPlanar(P_1, P_2|\alpha_{thresh}, d_{thresh})$ which decides whether two points P_1 and P_2 could be part of the same planar surface within a tolerance defined by the thresholds α_{thresh} and d_{thresh}.

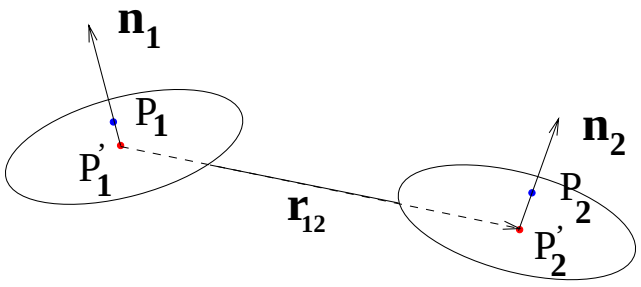

Figure 3.6: Coplanarity measure. Two planar patches fit around points P_1 and P_2 at a distance $|\mathbf{r_{12}}|$.

Sequential labeling

The previously defined metric of co-planarity and co-normality of two adjacent planar patches is what drives the cluster merging procedure. The cluster-merging is a sequential-labeling algorithm applied to a different domain. The original sequential-labeling algorithm [Ballard and Brown, 1982] is used to segment a binary image

into 8-connected segments of non-background pixels. We are doing something that is very similar. We are segmenting the range-image into 8-connected segments of 3–D points such that neighboring 3–D points have similar local planarity properties. Instead of having to decide whether two neighboring points have the same color, we have to decide whether two neighboring points are likely to be part of the same planar surface according to the metric introduced previously.

The cluster-merging algorithm works as follows. We visit all **locally planar** points in a raster-scan manner, starting from the upper-left corner of the range-image's grid (figure 3.2). Imagine that the algorithm is visiting point P (figure 3.7). Then we are considering P's three neighbors A_1, A_2 and A_3 (those four points belong to the clusters A, B, C and D respectively). If $CoPlanar(P, A_j|\alpha_{thresh}, d_{thresh})$ is TRUE, then the clusters where the two points P and A_j belong are merged.

In the implementation of the above algorithm, we do not need to merge each time the predicate is TRUE. Instead we can keep an equivalence table which holds equivalences between clusters. A second step is needed in order to resolve all equivalences and generate the final list $\mathbf{C}$ of clusters.

This algorithm has complexity $O(N)$ where N is the total number of range points (in our experiments $N = 10^6$).

3.4.3 Surface fitting

The next natural step is to fit a surface over the points of every cluster. A–priori knowledge of the types of surfaces that exist in the scene can be utilized in order to fit surfaces of the appropriate type. The most common surfaces encountered in urban scenes are planes. By fitting a plane over the points of each cluster a list

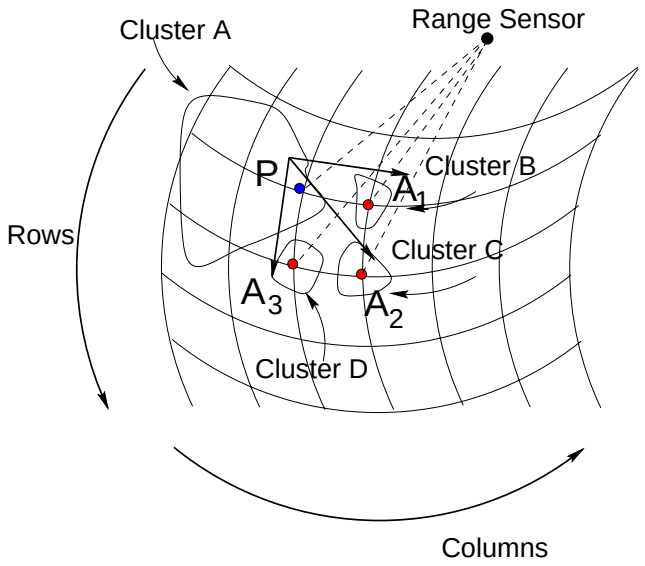

Figure 3.7: Sequential visit of initial clusters.

of planar surface is produced: $\mathbf{P} = \{\mathcal{P}(S_1), \ldots, \mathcal{P}(S_n)\}$. Each planar surface is represented with four parameters.

3.4.4 Boundary extraction

The final step of the segmentation process is the extraction of the boundary points of every cluster. A range-point belongs to the boundary of the cluster if at least one of its 8-neighbors is not a member of the cluster. The connectivity is defined over the two-dimensional grid which is displayed in figure 3.2. The outer boundary $\mathcal{O}(S_i)$ of the surface S_i is the one that encloses all range-points of the cluster, whereas the inner boundaries $\mathcal{I}_k(S_i)$ are holes inside the cluster (see figure 3.3).

Each boundary is thus a sequence of 2–D grid points:

$$\mathbf{B} = \{(i_0, j_0), (i_1, j_1), \ldots, (i_m, j_m)\}$$

Those grid points are ordered in the counter-clockwise direction. The actual 3–D

boundary is just:

$$\mathbf{B3D} = \{r(i_0, j_0), r(i_1, j_1), \ldots, r(i_m, j_m)\}$$

where $r(i, j)$ is the 3–D position which corresponds to the grid point (i, j).

We are also computing the 3–D axis-aligned bounding box **BOUND3D** of the set **B3D** (figure 3.8). The bounding box is used for a fast and rough estimation of the extention of the 3–D boundary in space and is being used in the algorithms for 3–D line extraction (section 3.5). So each boundary of a cluster is represented by the three sets: **B**, **B3D** and **BOUND3D**.

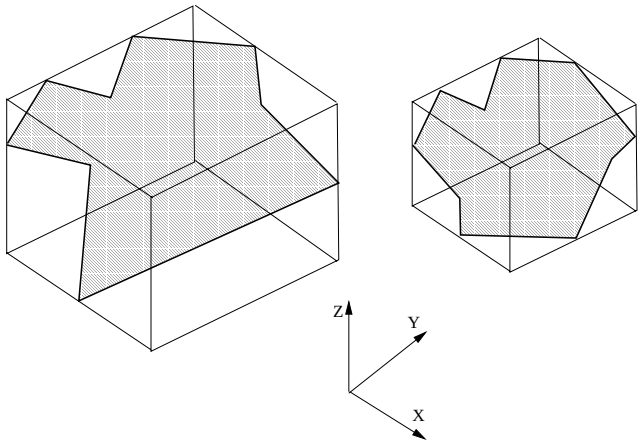

Figure 3.8: Axis-aligned bounding boxes of two planar 3–D surfaces.

3.4.5 Range segmentation summary

We presented the algorithm we developed for segmenting the range data-set into smoothly varying surfaces. In the first stage (**point classification**, section 3.4.1) we reject all points which can not be part of a local plane. In the second stage (**cluster merging**, section 3.4.2) large clusters of connected points are generated.

In the later stages, a planar surface is fit to the points (section 3.4.3) of each cluster and the boundary of this surface is extracted (section 3.4.4).

The algorithm is completely automatic. The user, though, has to provide three thresholds.

- The threshold P_{thresh} is used to decide whether a 3-D point can be part of a smoothly varying surface (**point classification** phase, section 3.4.1). Points, whose local planes produce fitting errors[2] above this threshold, are consider to lie on a local discontinuity. Thus, they are rejected from further consideration.

- The thresholds α_{thresh} and d_{thresh} are used to decide whether two planar patches have similar orientation and position in space (**cluster merging** phase, section 3.4.2). Local patches whose angular distance is greater than α_{thresh} or whose positional distance is greater than d_{thresh} are not considered to be parts of the same smoothly varying surface. Thus, by changing those thresholds we are able to achieve different segmentation results. No research was done in the direction of automatic threshold selection though (future work).

In the next section we present the 3–D feature extraction which is based on the segmented 3–D surfaces.

3.5 3–D feature extraction

At a second level of abstraction the 3–D range data-set is represented as a set of 3–D curves. Those curves are the result of intersection of neighboring bounded

[2] Minimum least squares errors.

3–D surfaces which have been extracted by the range segmentation module. In the context of this thesis we implemented the extraction of 3–D lines as a result of planar surface intersections. Those 3–D features are used for the registration between 3–D data-sets and between 3–D data-sets and 2–D images.

The extraction of 3–D lines involves three steps (figure 3.9):

1. Intersection of neighboring 3–D planes to produce 3–D lines of infinite extent (figure 3.10a).

2. Verification of the infinite 3–D lines. This step involves the computation of the distance between the bounded surfaces and the produced 3–D line (figure 3.10b).

3. Generation of 3–D linear segments out of the infinite 3–D lines. This is done by keeping the parts of the infinite 3–D lines which are verified by the range data-set (figure 3.10c).

Thus the 3–D line extraction algorithm works as follows:
For every pair (S_i, S_j) of neighboring 3–D surfaces generate the infinite 3–D lines $L(S_i, S_j)$ as the intersection of those surfaces. Then verify the existence of that line by computing its distance from the two surfaces S_i and S_j. Finally, extract the verified 3–D linear segment $LS(S_i, S_j)$ out of the infinite 3–D line. Each step is described in more detail in the following paragraphs.

3.5.1 Intersection of neighboring 3–D planes

The intersection of planes which are far with respect to each other produces fictitious lines which are not part of the range data set. Those fictitious lines can be

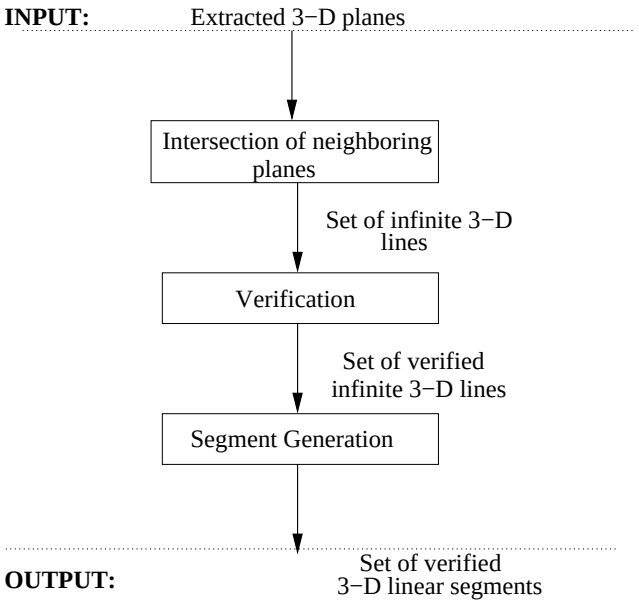

Figure 3.9: 3-D line extraction algorithm.

discarded at the second step of the algorithm. In order though to increase the time efficiency of the algorithm we use a measure of rough closeness between two planar surfaces.

Two bounded planar surfaces S_1 and S_2 are roughly close with respect to each other when their 3–D dimensional bounding boxes $\mathbf{BOUND3D}(S_1)$ and $\mathbf{BOUND3D}(S_2)$ (section 3.4.4) have a distance which is less than a user supplied threshold d_{bound}. So, we can define the following predicate:

$$RoughlyClose(S_1, S_2 | d_{bound}) = TRUE \Leftrightarrow$$
$$Distance(\mathbf{BOUND3D}(S_1), \mathbf{BOUND3D}(S_1)) < d_{bound}.$$

The distance computation between two axis-aligned bounding boxes is very fast. That means that this distance measure can be used in order to discard surfaces which are very far to each other in a fast manner. Intersections of 3–D planes is

performed only between planes S_1 and S_2 which make the $RoughlyClose$ predicate TRUE.

3.5.2 Verification of the infinite 3–D lines

The previous step generates a set of 3–D lines which may be the result of the intersection of non-neighboring planes since the predicate used is an approximation of the actual distance between two surfaces. In order to filter out the fictitious lines out of this set we introduce a second level of 3–D line verification. That is, we compute the distances D_1 and D_2 (figure 3.10b) of each 3–D line from the polygons which produced it. Then, we disregard all lines whose distance from both producing polygons is larger than a user-supplied threshold d_{poly} ($D_1 > d_{poly} \wedge D_2 > d_{poly}$).

The distance between a finite 3–D line and a polygon [3] is the **minimum** distance of this line from every edge of the polygon (that is true when the line does not pierce the polygon). In order to compute the distance between two line segments we use a fast algorithm described in [Lumelsky, 1985]. So, we define a measure of the distance between a line L and a polygon S_i when both the line and the polygon belong to the same plane:

$$LinePolygonDistance(L, S_i) = D_i.$$

3.5.3 Generation of 3–D linear segments

Finally we need to keep the parts of the infinite 3–D lines which are verified from the data set (that is we extract linear segments out of the infinite 3–D lines). We

[3] Both the line and the polygon lie on the same plane.

compute the distance between every point of the surfaces S_i and S_j and the line $L(S_i, S_j)$. We then create a list of the points whose distance from the line L is less than d_{poly} (see previous paragraph). Those points (points which are close with respect to the limit d_{poly} to the line) are projected on the line. The linear segment which is bounded by those points is the final result (see figure 3.10c).

3.5.4 3–D feature extraction summary

We presented an algorithm for extracting 3–D lines of finite extent from a segmented range data-set. The 3–D lines are the result of intersection of neighboring 3–D surfaces. The 3–D surfaces are very robustly extracted because they are supported by a large number of 3–D points. That transforms to robustly extracted 3–D lines. The accuracy of our 3–D lines is verified by our range to range and range to image registration experiments. The accurate registration between range and range data and between range and image data can not be possible with inaccurately localized 3–D lines.

3.6 Segmentation Results

The segmentation algorithms have been tested on range scans of urban structures. We have chosen four buildings. Two of them, the **Casa Italiana** and **Teacher's College Building** are part of the Columbia University Campus in New York City and are typical urban structures. These are buildings with planar façades and regular patterns of windows and doors. We also scanned the front of the **Guggenheim**

Museum in New York City[4], a one of a kind building with conical façades. Our final scanned building is the **Flat-Iron Building**, a trademark of New York's early century architecture.

The range and segmented scans of three views of **Casa Italiana** are shown in figures 3.11, 3.12 and 3.13. One view of **Teachers College** is shown in figure 3.14. Three views of **Guggenheim Museum** are shown in figures 3.16, 3.17 and 3.18. Finally, two views of the **Flat-Iron Building** are shown in figures 3.19 and 3.20. The range data-sets are presented on the top of each figure and the segmented results on the bottom. Each segmented surface is displayed with different color. The points which failed the initial classification step (section 3.4.1) are displayed with red color.

The segmentation algorithm correctly extracts planar regions. In the case of Casa Italiana, all major walls have been extracted as well as small bricks and window borders. The same is true in the case of Teachers College, where we are able to extract parts of the roof and window shades. The first view of Flat-Iron building has been segmented into two major planar regions and a large number of window borders have been identified. In the second view one major wall has been extracted. Finally, in the case of Guggenheim Museum the segmentation algorithm is able to extract conical façades. Thus, the algorithm can extract slowly varying smooth surfaces and not exclusively planes. This is because in the cluster merging phase we are using a local region-growing decision which does not force the extracted regions to lie on a plane.

Figure 3.15 displays the results of the line extraction algorithm for the first

[4]Designed by Frank Lloyd Wright, one of the most famous architects of the 20^{th} century.

views of the Casa Italiana and Teachers College. It is clear that major linear features (borders of large walls) as well as borders of windows have been extracted correctly.

Table 3.1 displays the parameters used for the presented segmentation results. This table contains the size $k \times k$ of the neighborhood used to fit the initial local planes (section 3.4.1) and the planarity threshold P_{thresh} used to reject points where local planes cannot reliably fit (section 3.4.1). It also contains the co-planarity and co-normality thresholds d_{thresh} (in meters) and α_{thresh} (in degrees) used to grow the initial regions of locally planar points (section 3.4.2).

Table 3.2 presents the parameters used for 3–D line detection. Those are the distances d_{bound} and d_{poly} (in meters) used to verify the line segments which are produced from the intersection of the segmented surfaces (sections 3.5.2 and 3.5.3). Finally, table 3.3 contains the sizes of each data-set.

Segmentation Parameters				
Building	$k \times k$	P_{thresh}	α_{thresh}	d_{thresh}
Casa Italiana Scans 1 and 2	7×7	0.08	0.04^o	0.1 m
Scan 3	7×7	0.08	0.03^o	0.08 m
Teachers College	7×7	0.08	0.04^o	0.1 m
Guggenheim Museum Scan 1	7×7	0.01	0.8^o	0.05 m
Scans 2 and 3	7×7	0.008	0.4^o	0.01 m
Flat-Iron Building Scan 1	5×5	0.08	0.1^o	0.01 m
Scan 2	7×7	0.08	0.075^o	0.02 m

Table 3.1: Parameters used for range segmentation

3-D line extraction parameters		
Building	d_{bound}	d_{poly}
Casa Italiana	0.5 m	0.3 m
Teachers College	0.4 m	0.2 m
Guggenheim Museum	N/A	N/A
Flat-Iron Building Scan 1	0.5 m	0.2 m
Scan 2	0.4 m	1.0 m

Table 3.2: Parameters used for 3–D line detection

Data-set sizes	
Building	$N \times M$ points
Casa Italiana Scan 1	963 by 940
Scan 2	589 by 727
Scan 3	926 by 937
Teachers College	992 by 998
Guggenheim Museum Scan 1	957 by 907
Scan 2	612 by 604
Scan 3	502 by 502
Flat-Iron Building Scan 1	773 by 881
Scan 2	962 by 969

Table 3.3: Data set sizes

3.7 Threshold sensitivity

In this section we qualitatively describe the effect of the five user-specified thresholds on the segmentation and 3-D line extraction routines (P_{thresh}, α_{thresh}, d_{thresh}, d_{bound} and d_{poly}).

In the segmentation phase, the threshold P_{thresh} is used for the identification of non-planar points, that is points which lie on surface discontinuities. In our experiments P_{thresh} is ranging from 0.008 to 0.08. The smaller the threshold P_{thresh} the larger the number of points which are classified as lying on surface discontinuities. A small P_{thresh} results to one large cluster of un-segmented points and to a large number of small clusters of segmented points. On the other hand a large value

for P_{thresh} will result to a small number of surface discontinuities with the possible drawback of over-segmentation, depending on the selection of the thresholds α_{thresh} and d_{thresh}. This is because cluster-merging ends if

1. A surface discontinuity has been reached[5], or
2. The co-planarity or co-normality metrics are smaller than the values of α_{thresh} and d_{thresh}.

The output of the segmentation algorithm is sensitive on the thresholds α_{thresh} (from 0.03^o to 0.8^o in our experiments) and d_{thresh} (from 0.01 m to 0.1 m in our experiments). Large values for both thresholds will result in a small number of large clusters (under-segmentation) whereas small values for both threshold will result in a large number of small clusters (over-segmentation). In the extreme cases, if infinite values for the thresholds are used the algorithm will classify the locally planar points into maximum connected components, whereas if zero values are used no cluster merging will be performed.

In the 3-D line extraction phase, the threshold d_{bound} is used for a crude approximation of spatial closeness between segmented surfaces (from 0.4 to 0.5 m in our experiments). A small value for d_{bound} may lead in discarding a number of valid lines, whereas a large value will not hurt the correctness, but only the efficiency of the algorithm. The value of threshold d_{poly} (from 0.2 to 1.0 m in our experiments) is more important. A very small value will lead to the detection of a small number of lines (even to the detection of no lines at all), whereas a large value can result to the detection of large number of fictitious lines[6] due to the intersection

[5]Only locally planar points can be part of a segmented cluster.

[6]Fictitious lines however do not affect the matching procedures of chapter 5.

of non-neighboring planes.

3.8 Conclusions

We developed a general range-segmentation and 3–D feature detection algorithm. This algorithm is applicable to scenes which are densely scanned. We proved the applicability of these algorithms in scenes of urban structures. In our results we showed that we can extract large planar surfaces, small bricks and window borders as well as conical sections. The segmentation algorithms are automatic. The user, though, has to specify thresholds which tune the segmentation for particular data-sets.

The segmentation algorithms are very efficient. The surface extraction algorithm has a complexity of $O(N)$ where N is the total number of range points. The feature extraction algorithm has a complexity of $O(M^2)$ where M is the number of extracted 3–D planes.

The extracted surfaces are used for efficient 3–D modeling (chapter 4) and the extracted 3–D features for range to range and range to image registration (chapter 5).

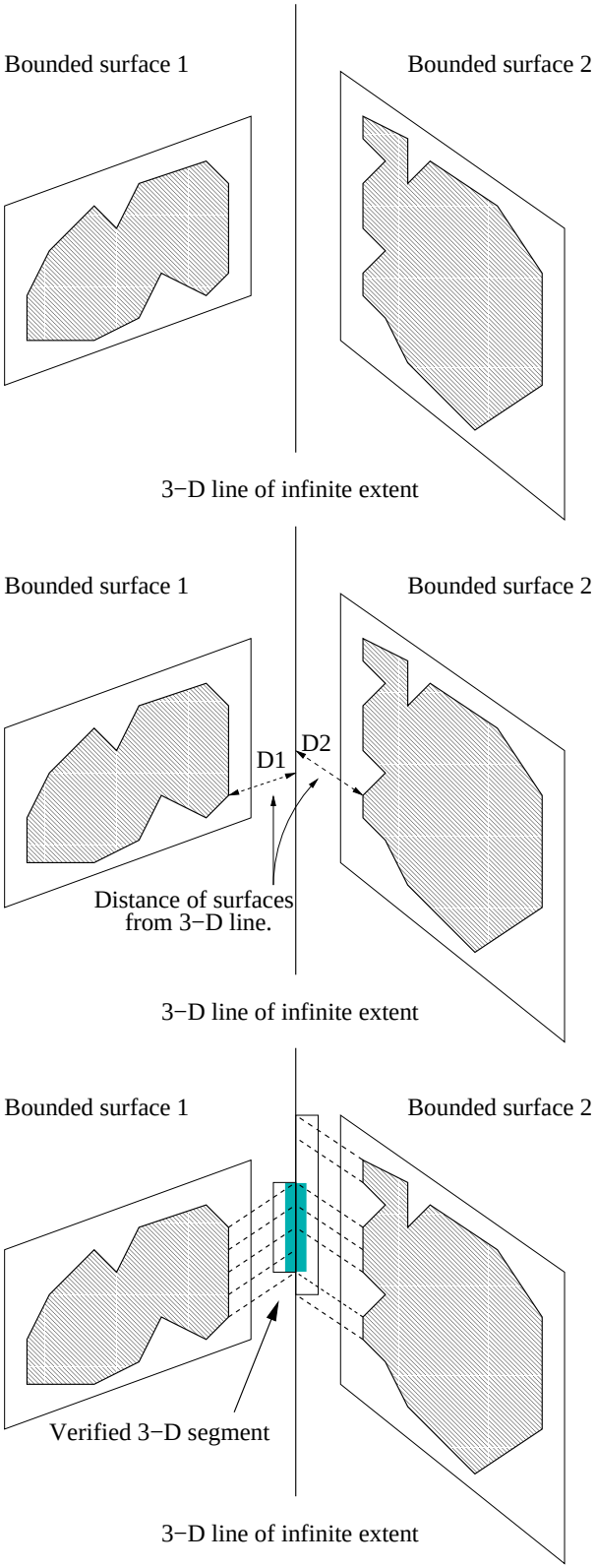

Figure 3.10: a) Infinite 3–D line as intersection of neighboring planes, b) distance of bounded surfaces from line and c) verified 3–D linear segment.

Figure 3.11: **Casa Italiana**. Range and segmented scans (first view). 976 × 933 points.

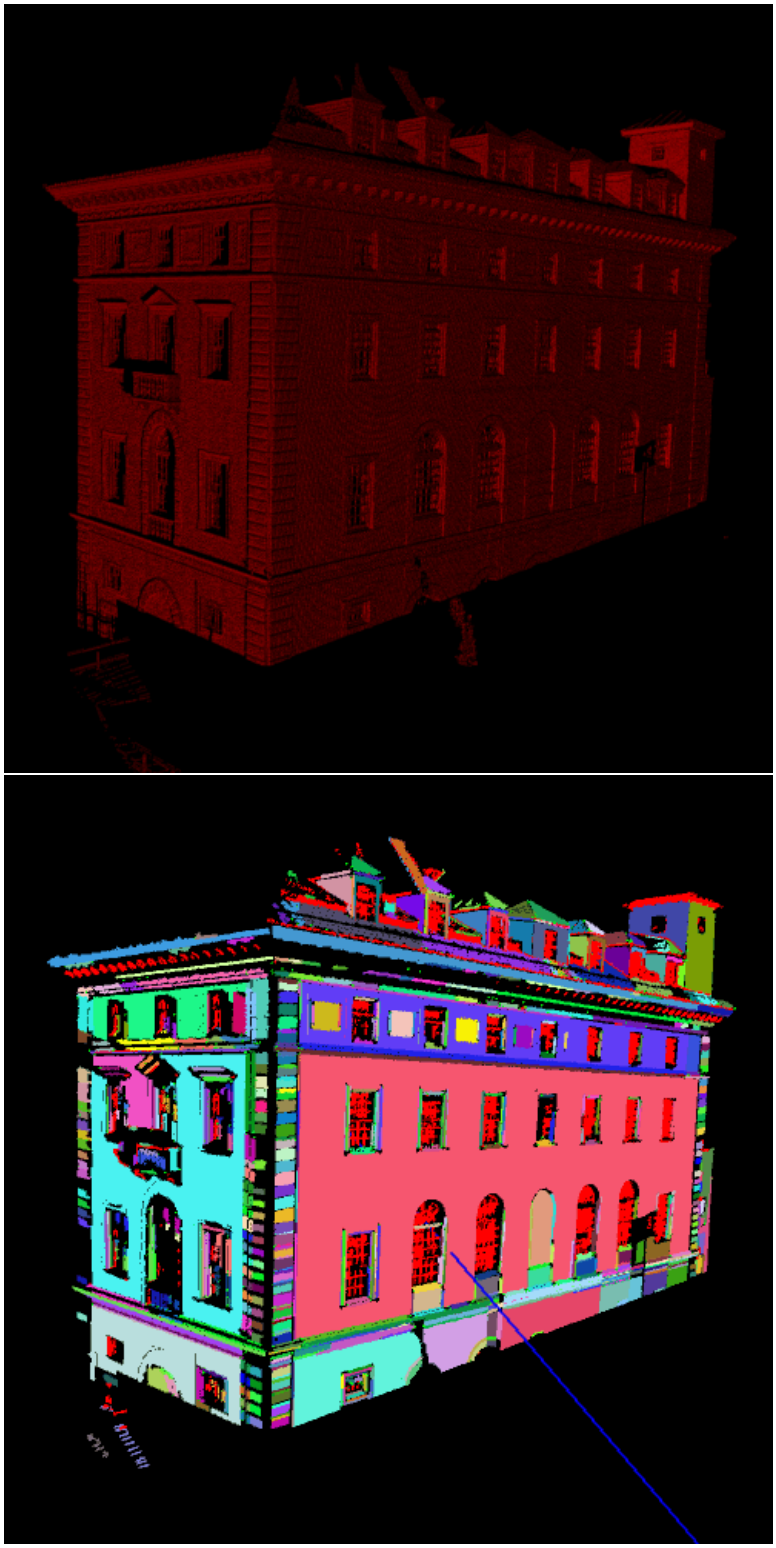

Figure 3.12: **Casa Italiana**. Range and segmented scans (second view). 589 × 727 points.

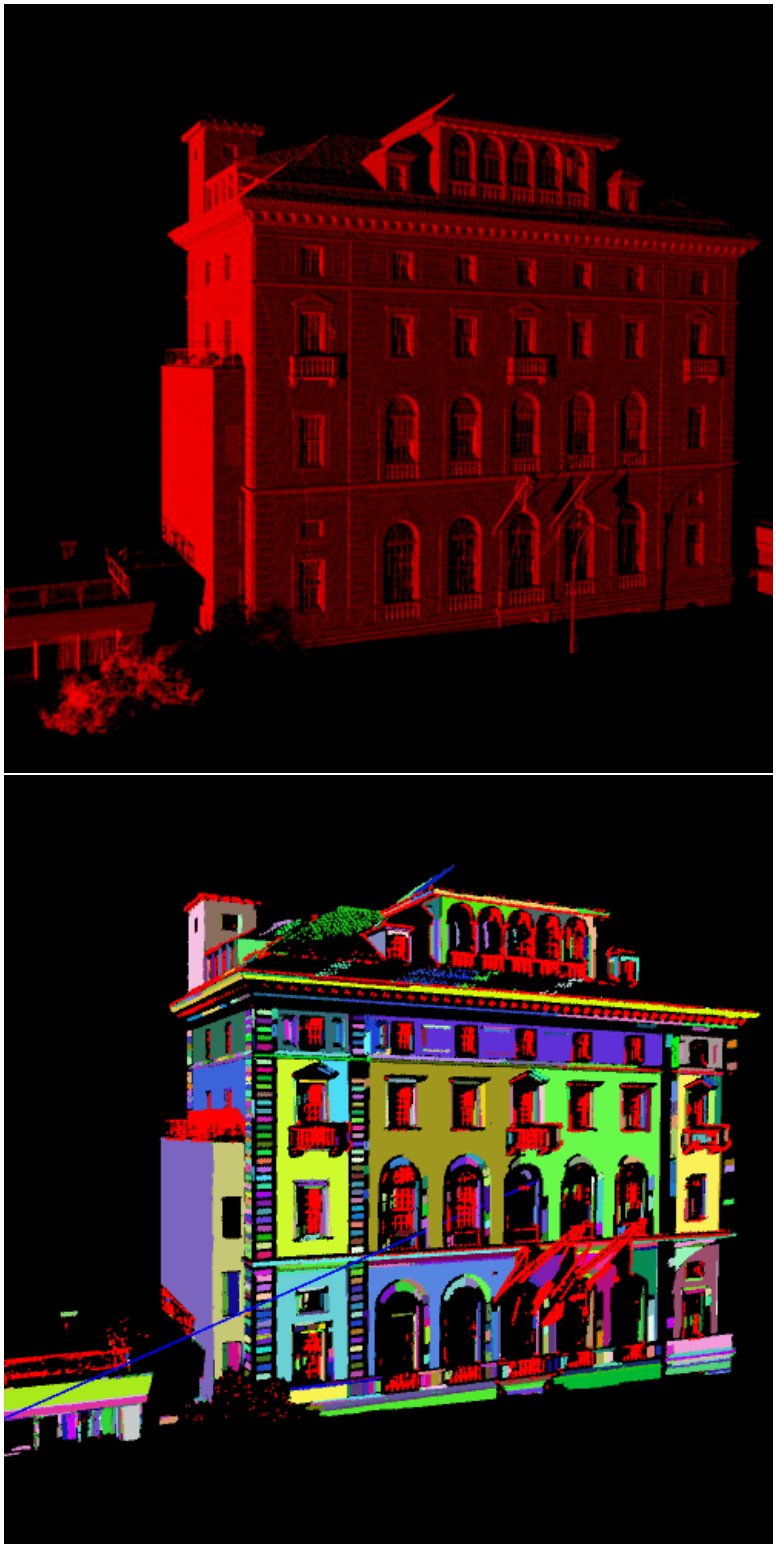

Figure 3.13: **Casa Italiana**. Range and segmented scans (third view). 926 × 937 points.

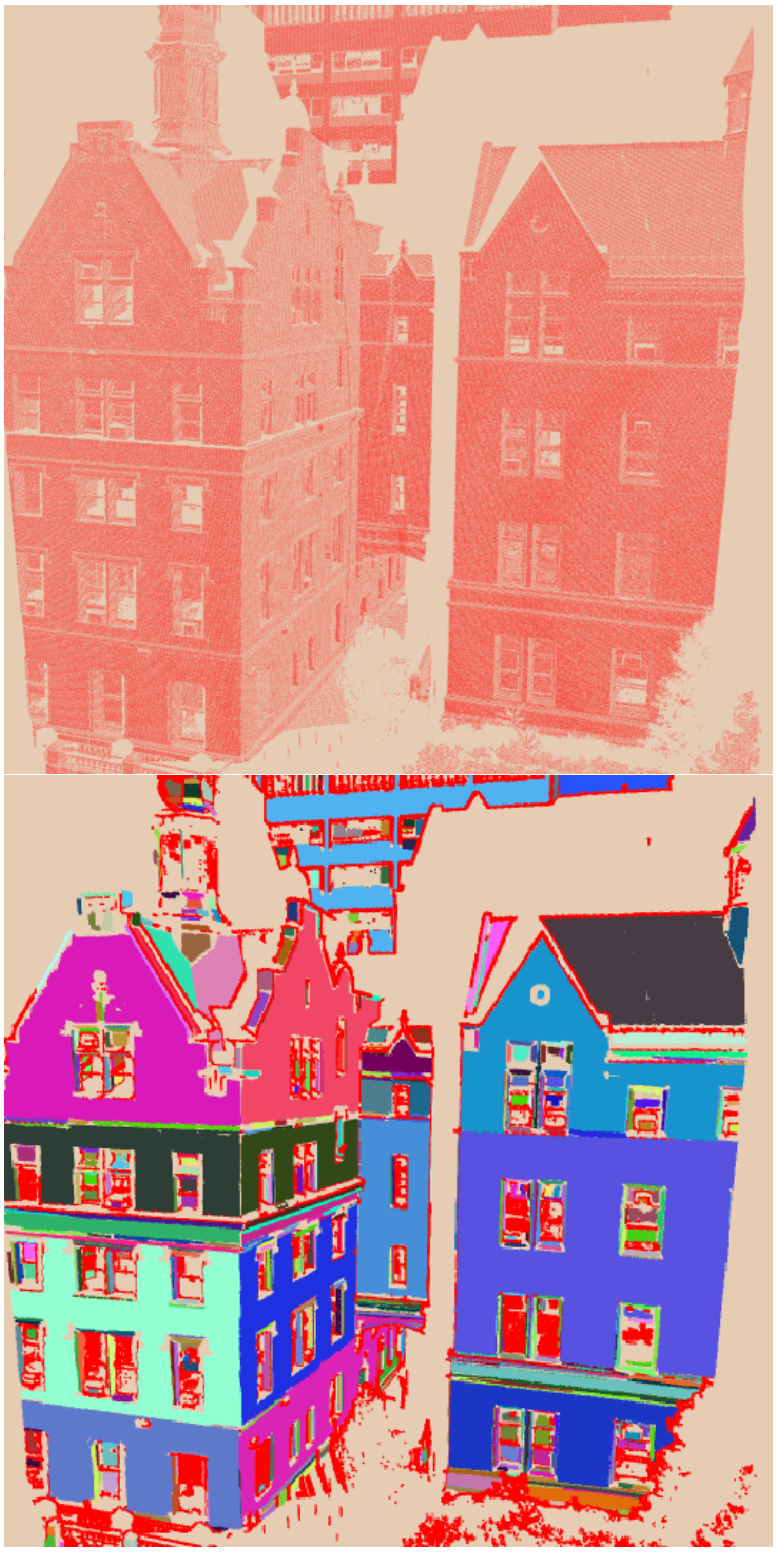

Figure 3.14: **Teachers College**. Range and segmented scans (first view). 992 × 998 points.

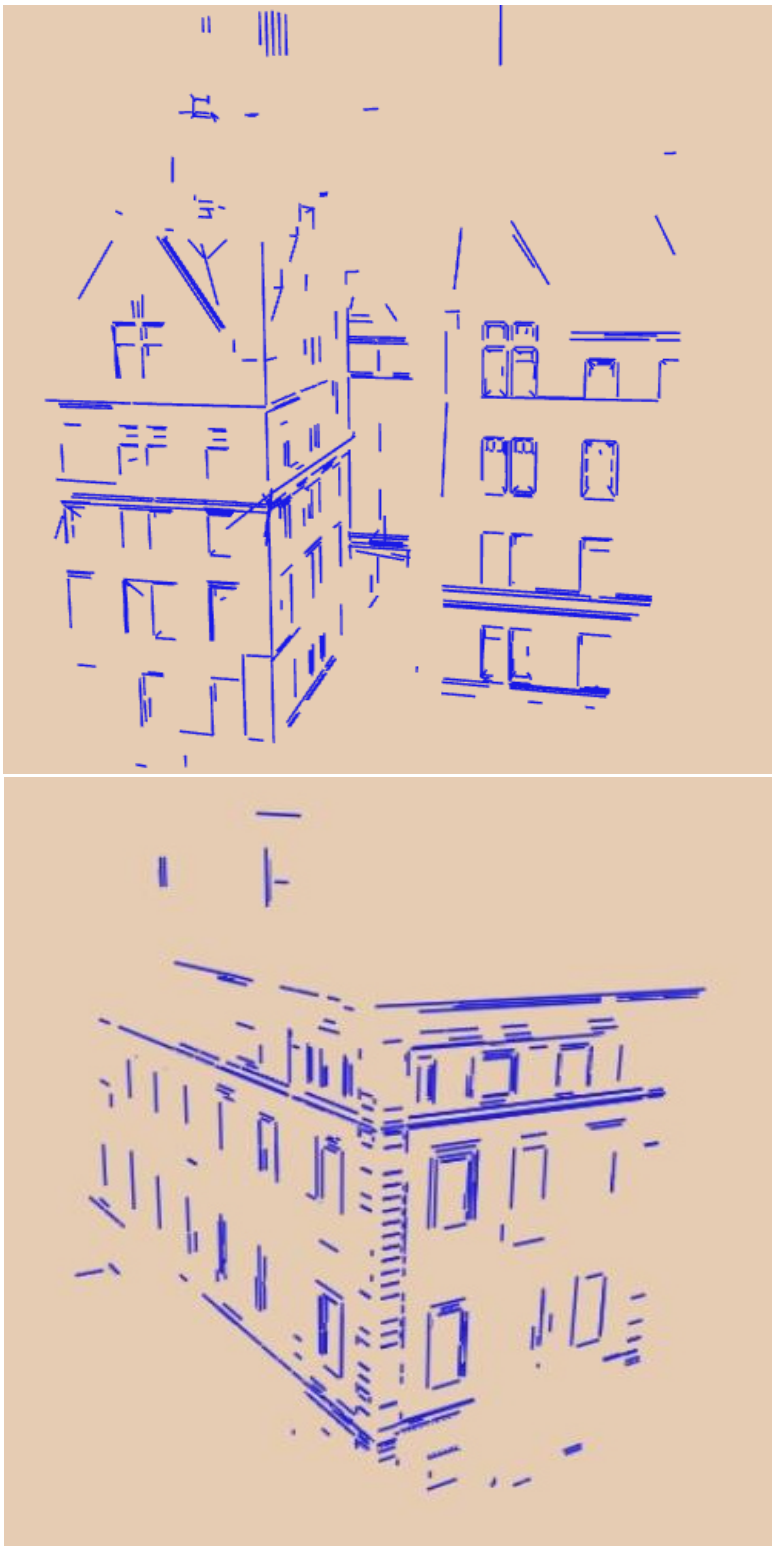

Figure 3.15: 3-D lines. Teachers College and Casa Italiana (first view).

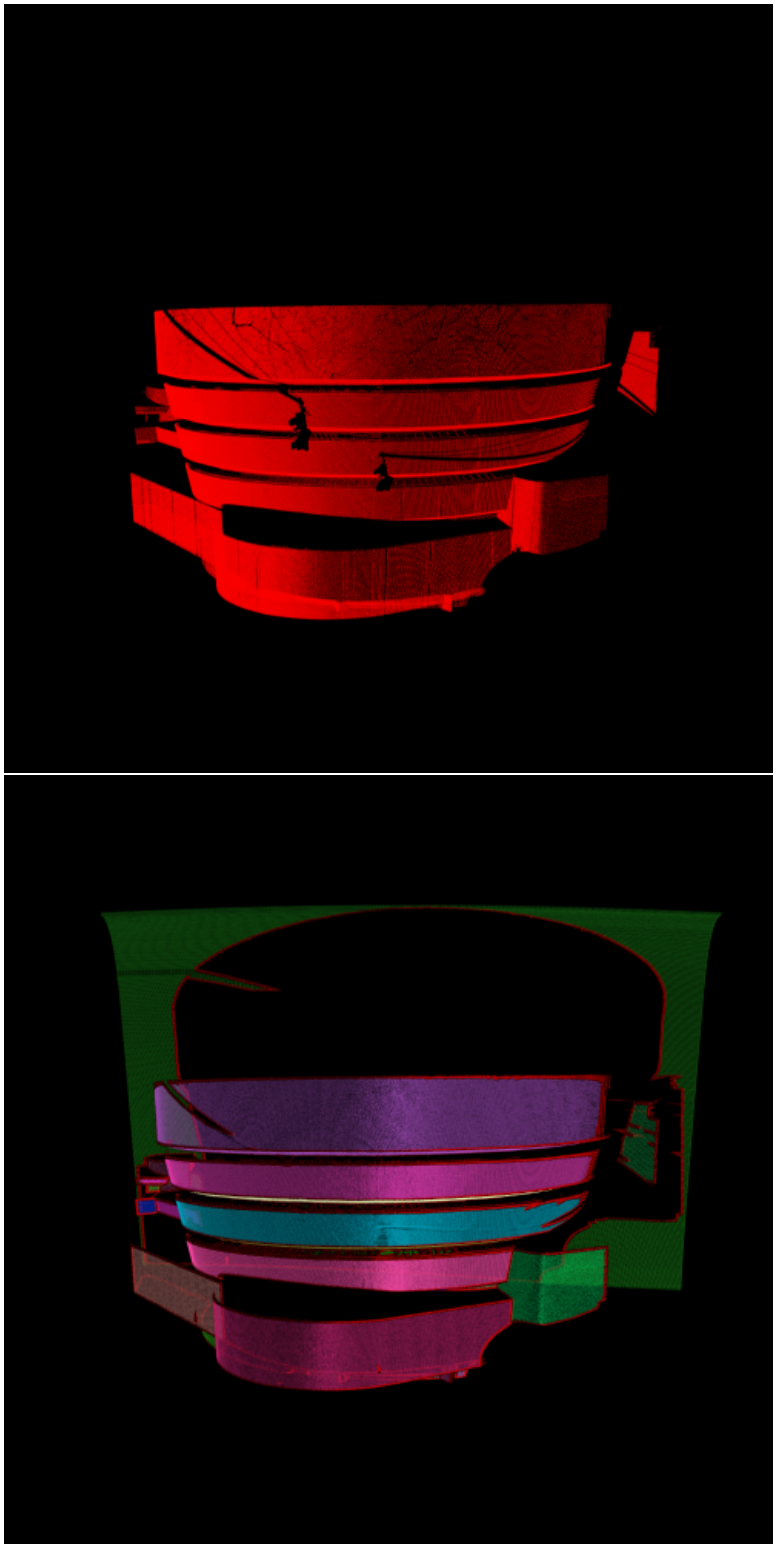

Figure 3.16: **Guggenheim Museum**. Range and segmented scans (first view). 957×907 points.

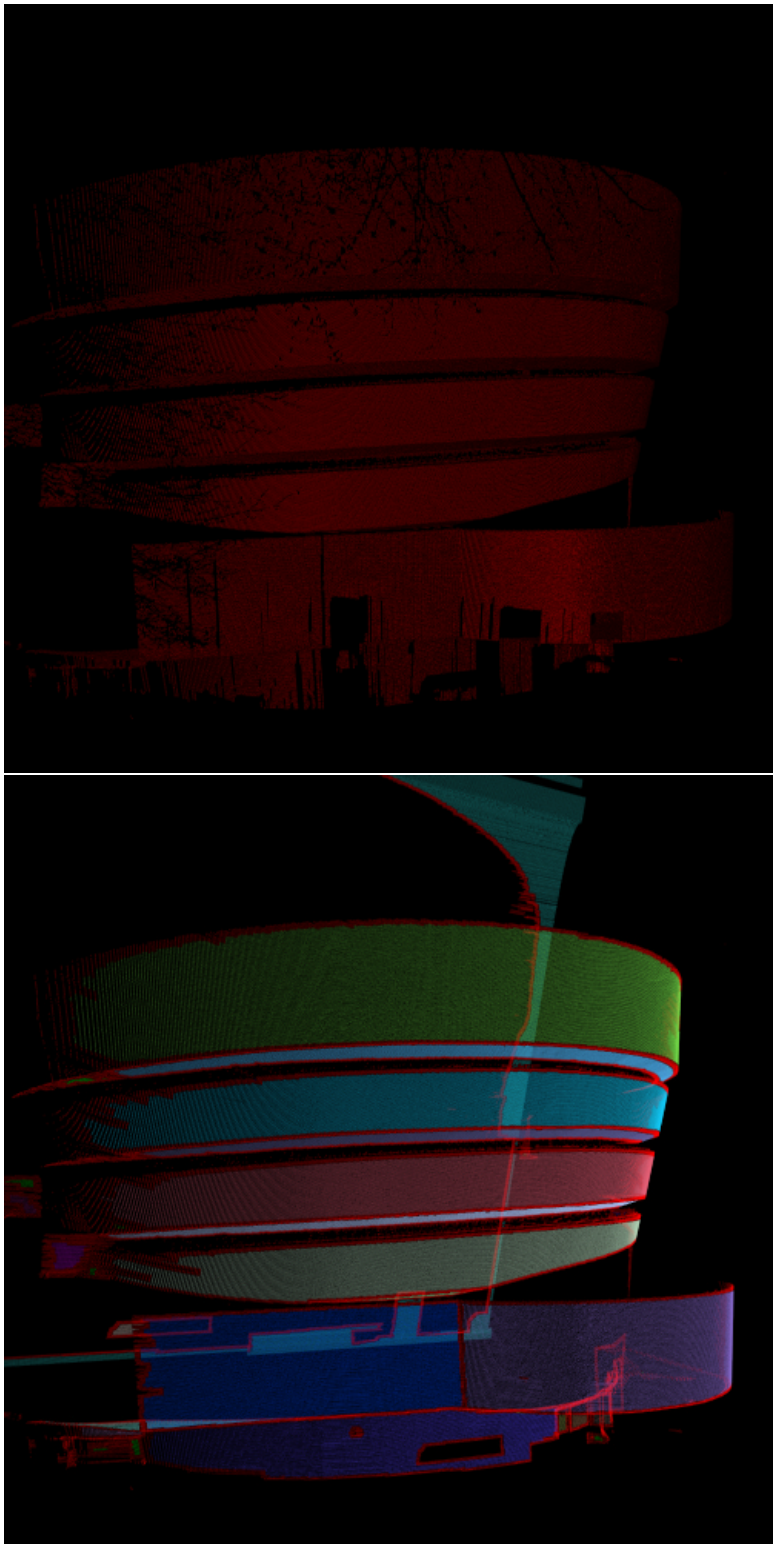

Figure 3.17: **Guggenheim Museum**. Range and segmented scans (second view). 612×604 points.

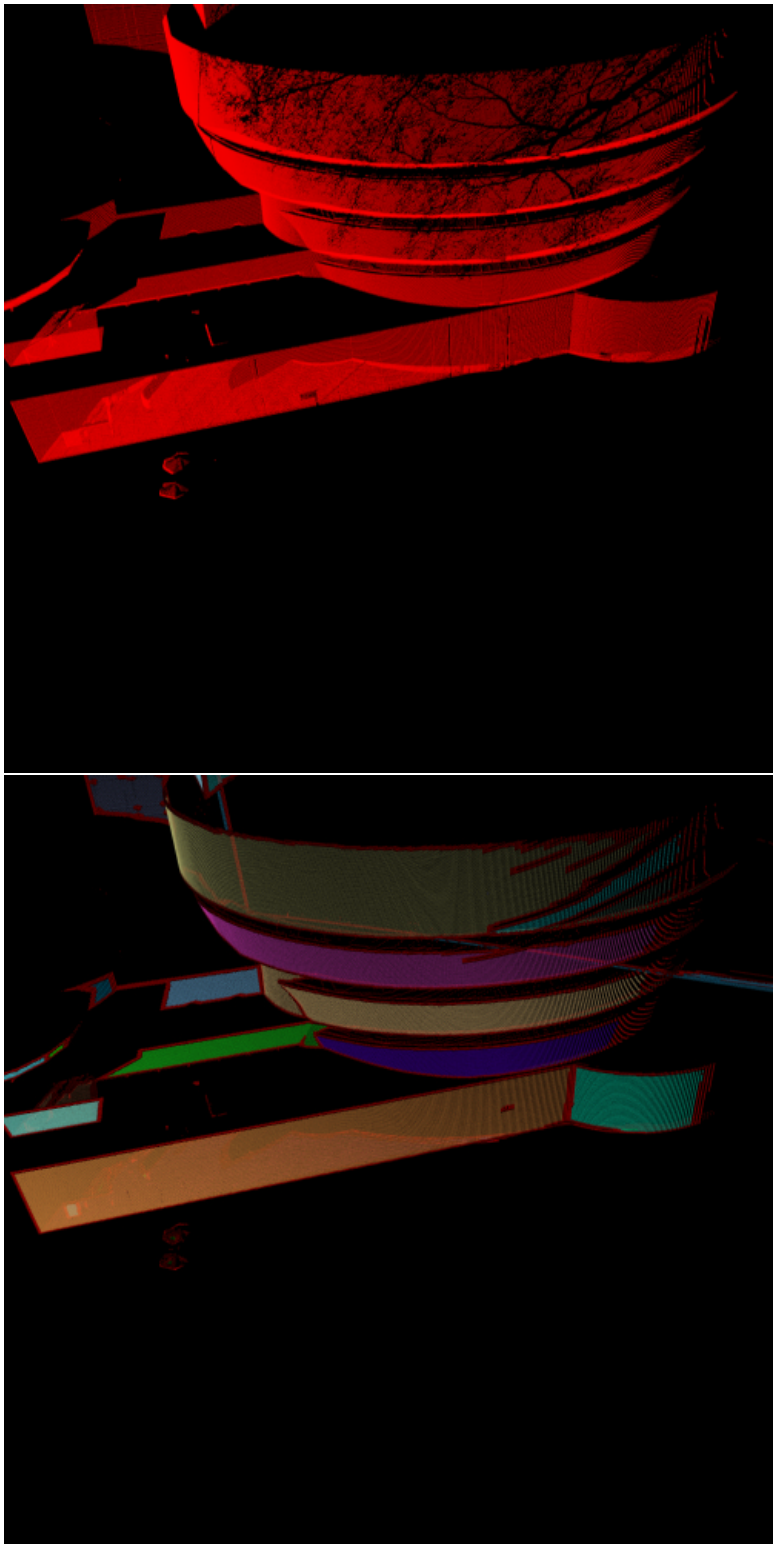

Figure 3.18: **Guggenheim Museum**. Range and segmented scan (third view). 502×502 points.

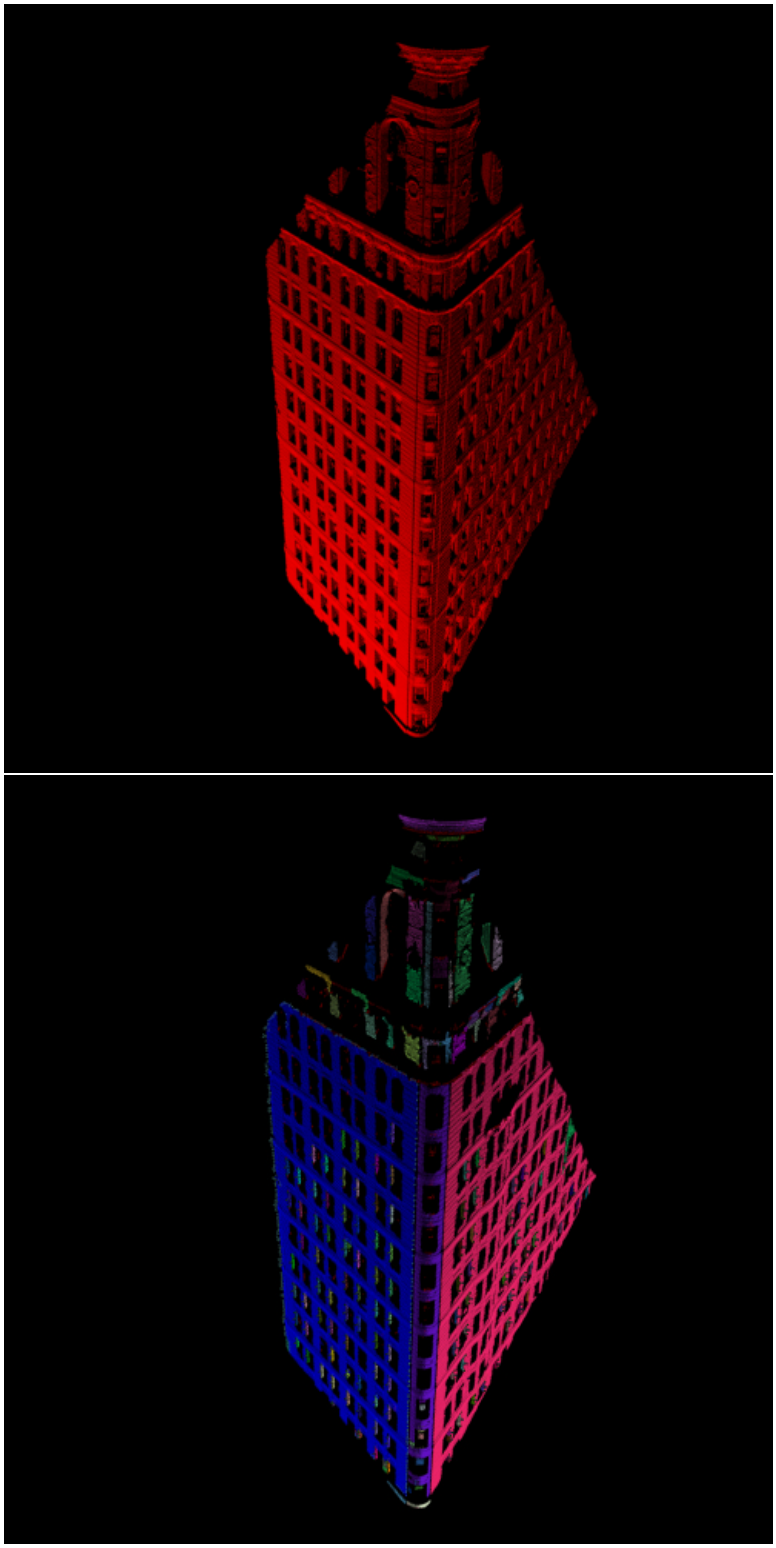

Figure 3.19: **Flat-Iron Building**. Range and segmented scan (first view). 773 × 881 points.

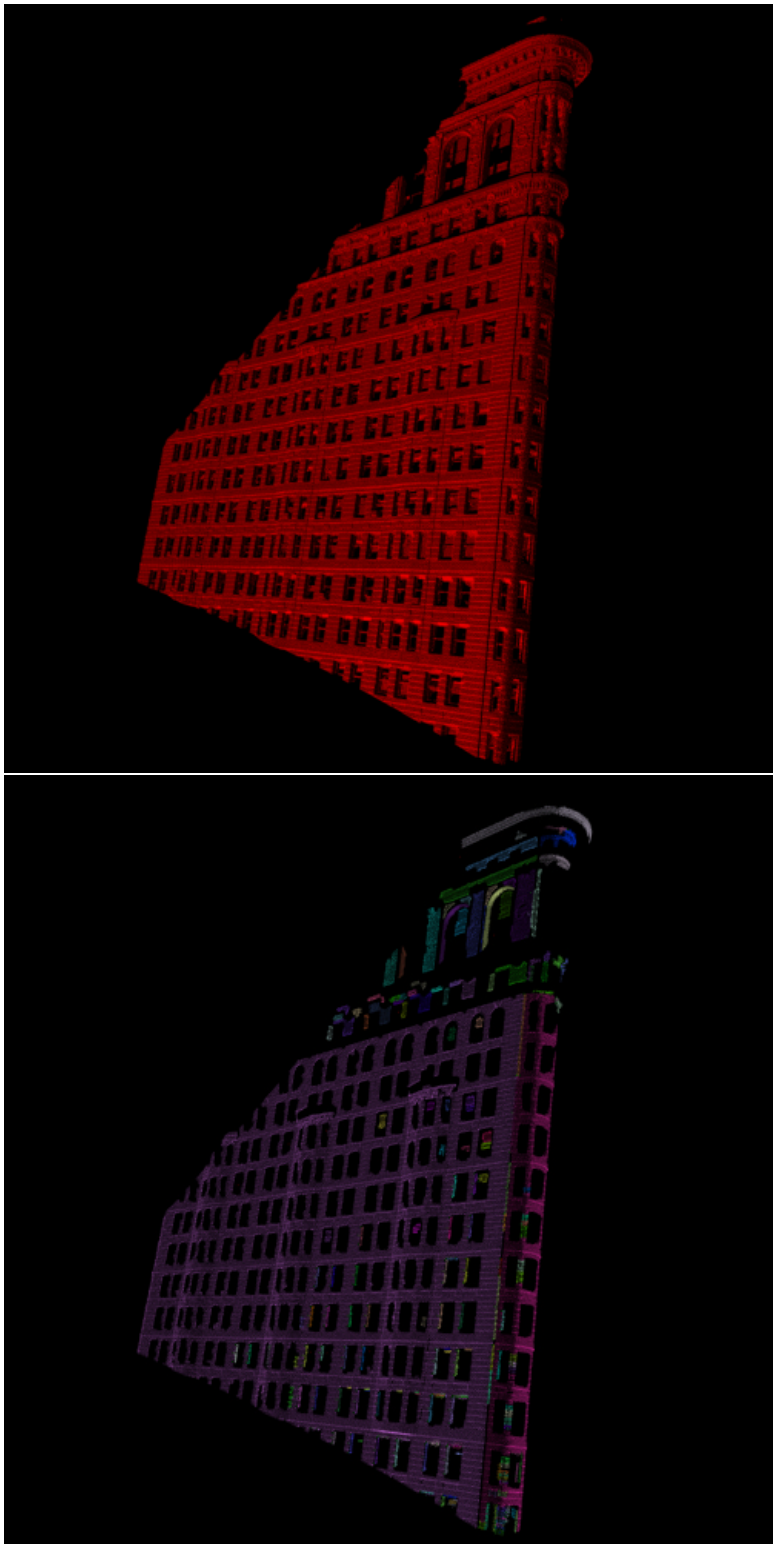

Figure 3.20: **Flat-Iron Building**. Range and segmented scans (second view). 962 $\times$ 969 points.

Chapter 4

Solid Modeling

4.1 Introduction

This chapter deals with the construction of solid CAD models of large urban structures from range measurements. In order to capture the complete geometric information of a large outdoor scene, the scene needs to be sensed from a number of different viewing directions. That produces a set of complex and overlapping 3-D point clouds. Our goal is the utilization of all point clouds and the creation of a volumetric representation which captures the visible surface of the scene. Such representations, enhanced with photometric observations, provide efficient, geometrically correct and photometrically accurate visualizations from all possible viewpoints which surround the scene. Volumetric representations also provide an intuitive framework for representing and filling holes created from self-occlusion or from limitations of the range sensing technology[1].

A typical solid modeling system involves the phases of

[1]Range sensors fail to measure the distance of transparent objects, such as windows.

1. Individual range image acquisition from different viewpoints.

2. Registration of all images into a common frame of reference.

3. Transformation of each range image into an intermediate surface-based or volumetric-based representation.

4. Merging of all range images into a common representation.

In general, 3-D modeling systems are based either on mesh-based or volumetric approaches. In the mesh-based approaches, each range image is transformed into a mesh of triangular faces and all range images are being merged by the averaging of surface elements on the mesh level. The final result is a triangular mesh which approximates the outer surface of the sensed object ([Turk and Levoy, 1994] is a representative approach). Volumetric approaches, on the other hand, combine individual range images into a 3-D volume. The underlying representation is the 3-D volume which approximates the actual volume the sensed object occupies (see [Curless and Levoy, 1996, Reed and Allen, 1999] for representative approaches). Volumetric approaches are considered superior to mesh-based methods, since they can model and fill holes in the final models. Holes can destroy the photorealistic appearance of the scene and thus are highly undesirable.

Large urban scenes drive existing solid modeling systems to their limits since high resolution and dense range scanning is needed to capture small architectural detail in buildings, resulting in data-sets of high complexity. The ability to simplify the acquired data set before the integration to the final model is equivalent to an adaptive sampling of the scene. Areas of high geometrical complexity are densely sampled, whereas areas of smoothly varying surfaces (e.g. planes) are represented

by a fewer number of samples.

Most volumetric solid-modeling methods use the full input-data set without performing any significant simplification to it. An example is the work of Curless and Levoy [Curless and Levoy, 1996]) which does not provide any framework for simplification while the range data is being integrated into the model.

We have developed a volumetric solid-modeling system with the enhanced ability for data simplification in the intermediate modeling steps. This data simplification and segmentation of each individual range image *before* merging translates to simplified intermediate volumetric representations. Merging those intermediate representations is the *hardest* part of modeling. Thus, the decreased complexity of the individual volumes results in increased efficiency in the solid modeling phase. This is exactly the kind of efficiency that is highly desirable in the context of modeling large outdoor scenes. The method scales well with the increased sampling density of measured 3-D scenes and is thus appropriate for handling the complexity of scenes of large scale.

Our modeler is an extension to the solid modeler developed by Michael K. Reed [Reed and Allen, 1999, Reed *et al.*, 1997]. The key extensions are described in the sections below.

4.2 Original system

The innovative principle of Reed's approach is the representation of each individual range image with a solid volume. Each cloud of range points is being transformed to a triangular surface mesh, and that mesh to a solid volume. The volumes elegantly capture the sensed and "yet to be explored" parts of the scene. The difficult problem

of merging individual registered range images transforms to the computation of boolean CAD intersections between the partial solid volumes which represent each individual view of the scene.

The original model acquisition system constructs a solid CAD model incrementally using one range image at a time. For each range image a solid CAD model is being constructed. This model consists of sensed and extruded (along the sensing direction) surfaces. Sensed surfaces are the boundary between the inside and the outside (towards the sensor) regions of the sensed object, whereas extruded surfaces are the boundary between empty sensed 3-D space and un-sensed 3-D space. Extruded surfaces are used in order to plan for new sensor locations: what lies "behind" them has not been explored yet. The modeling concept is described in figure 4.1. A triangular surface mesh 4.1b is constructed from the raw range image of the sensed object 4.1a. That mesh is swept towards the sense direction and the resulting solid model is shown in 4.1c. Figure 4.2 explains what is happening in more detail. A part of the reconstructed triangular mesh of an object is shown in 4.2a. Each triangular mesh element is extruded towards the sense direction creating a triangular prism (4.2b). The final swept volume if the result of the boolean unification of the individual triangular prisms (4.2c).

The final solid model which captures the volume of the sensed scene is the result of the boolean intersection among the partial solid models which describe each individual view of the scene. Thus, the final solid representation of the scene is computed incrementally by the intersection of individual solid models which capture the scene. The characterizations of the surfaces (sensed or extruded) propagate from the individual models to the merged model. Therefore this method has the ability

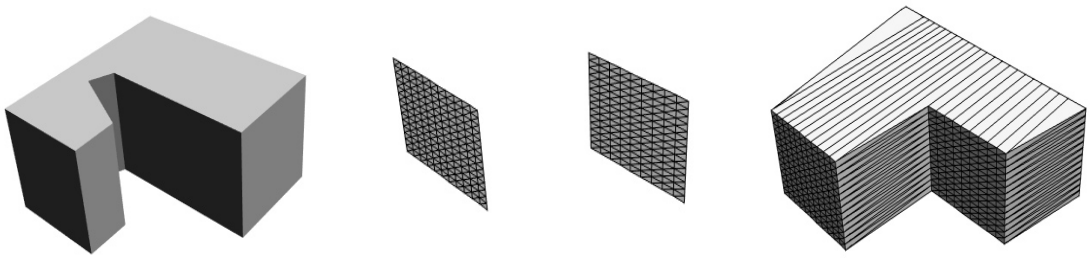

Figure 4.1: Solid Modeling concept introduced by Michael K. Reed (images printed after permission of the author). a)(left) Solid block being sensed, b) Mesh generated by the measured 3-D points, c) Partial solid model construction.

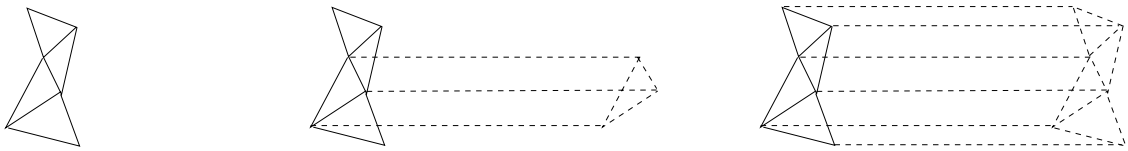

Figure 4.2: Solid Modeling concept introduced by Michael K. Reed. Sweeping triangular mesh elements. a)(left) Triangular mesh surface, b) One triangular element swept towards sensing direction, c) All triangular elements swept.

to capture surfaces of arbitrary shape, to incrementally include range images in the final representation and to include the range images in an order-independent way.

4.3 Extending the System for Large Outdoor Structures

In order to extend the system to outdoor environments we need to provide a method of registering individual range scans since we can't rely on devices of high positional accuracy. However, position estimates provided by the GPS system can be a very good initial guess. Also since urban scenes provide very reliable 3–D linear features, we decided to use those linear features for range registration algorithm. Another issue which we need to consider is the fact that parts of the model may be missing

due to transparent objects in the scene (such as windows). Those parts need to be interpolated from neighboring samples. Finally, the simplification of the dense mesh surfaces becomes important for efficient 3D modeling, efficient rendering of the scene and efficient application of the planning modules.

4.3.1 Range Data Registration

To create a complete description of a scene we need to acquire and register multiple range images. The registration (computation of the rotation matrix R and translation vector $\mathbf{T}$) between the coordinate systems of the n_{th} (C_n) and first (C_1) range image is possible via a matched set of 3–D features between the images. We have decided to use the infinite 3–D lines which are extracted using the algorithm described in chapter 3 as our features of interest. A manual match between a small number of those features provides enough constraints that can lead in the computation of the rotation R and translation $\mathbf{T}$.

In detail the algorithm works as follows. The infinite 3-D lines that are automatically extracted from the dataset[2] can be represented by the pairs of the form $(\mathbf{n}, \mathbf{p})$, where $\mathbf{n}$ is the unit vector which corresponds to the direction of the line and $\mathbf{p}$ is a 3-D position which represents a point on the line. Note that this representation is not unique. There are two valid line directions $\mathbf{n}$ and $-\mathbf{n}$ and an infinite number of points $\mathbf{p}$ that lie on the line. We choose $\mathbf{p}$ to be one of the extracted endpoints of the line.

A solution for the rotation and translation is possible when at least two line matches are given. The rotation matrix R can be computed according to the closed

[2]Our modules extract 3-D lines of finite extent. However, the extracted positions of the endpoints are not used for registration purposes because of the uncertainty in their determination.

form solution described in [Faugeras, 1996], page 523.

Lets assume that the lines

$$(\mathbf{n_i}, \mathbf{p_i}), i = 1 \ldots N$$

extracted automatically from one view do match up with the automatically extracted lines

$$(\mathbf{n_i}', \mathbf{p_i}'), i = 1 \ldots N$$

of the second view.

The rotation component of the transformation between the two view can be computed using the orientations $\mathbf{n_i}$ and $\mathbf{n_i}'$ of the matched 3-D lines. This is done via the minimization of the error function

$$Err(N) = \sum_{i=1}^{N} ||\mathbf{n_i}' - R\mathbf{n_i}||^2$$

where R is the unknown rotational matrix. The minimization of the above function has a closed-form solution when the rotation is expressed as a quaternion. The minimum number of correspondences for the computation of the rotation is two ($N = 2$). More lines can though be used in order to increase the robustness of the method. Note that in the above formulation we assume that we have a correspondence between the directed vectors $\mathbf{n_i}$ and $\mathbf{n_i}'$. Otherwise the minimization would be formulated as

$$Err(N) = \sum_{i=1}^{N} ||\epsilon_i'\mathbf{n_i}' - \epsilon_i R\mathbf{n_i}||^2$$

where $\epsilon_i, \epsilon_i = \pm 1$. The latter formulation results in 4 possible solutions for the rotation matrix. However, knowledge of the matching directions reduces the number of solutions to one.

Solving for the translation vector $\mathbf{T}$ between the two views is an easy task as long the rotational matrix has been computed. Let us select two arbitrary points on the i_th line $< \mathbf{n_i}, \mathbf{p_i} >$ of the first view. Those points can be expressed as $\mathbf{a_1^i} = \mathbf{p_i} + t_1\,\mathbf{n_i}$ and $\mathbf{a_2^i} = \mathbf{p_i} + t_2\,\mathbf{n_i}$ where $t_j, j = 1,2$ are two arbitrary real constants. Those two points have corresponding points which lie on the i_th line $< \mathbf{n_i}', \mathbf{p_i}' >$ of the second view. If we call those points $\mathbf{a_1^{i'}}$ and $\mathbf{a_2^{i'}}$ we can similarly express them as $\mathbf{a_1^{i'}} = \mathbf{p_i'} + t_1'\,\mathbf{n_i'}$ and $\mathbf{a_2^{i'}} = \mathbf{p_i'} + t_2'\,\mathbf{n_i'}$, where $t_j', j = 1,2$ are two real constants which depend on the arbitrary selection of $t_j, j = 1,2$. That means that the correspondence between the i_th lines of the first and second view provide us the following constraints:

$$\mathbf{a_1^{i'}} = R\mathbf{a_1^i} + \mathbf{T} \tag{4.1}$$

$$\mathbf{a_2^{i'}} = R\mathbf{a_2^i} + \mathbf{T} \tag{4.2}$$

or

$$\mathbf{p_i'} + t_1'\,\mathbf{n_i'} = R(\mathbf{p_i} + t_1\,\mathbf{n_i}) + \mathbf{T} \tag{4.3}$$

$$\mathbf{p_i'} + t_2'\,\mathbf{n_i'} = R(\mathbf{p_i} + t_1\,\mathbf{n_i}) + \mathbf{T} \tag{4.4}$$

When the rotation matrix R is known the above system of 6 equations is linear in the 7 unknowns (3 for the translation vector $\mathbf{T}$ and 4 for the real constants t_j and $t_j', j = 1,2$). With two line matches the number of equations becomes 12 ($= 2 \times 6$) and the number of unknowns 11 ($= 2 \times 4 + 3$). That means that a minimum of two matched infinite 3-D lines provide enough constraints for the computation of the translation (when the rotation is known) through the solution of an over-constrained system of linear equations.

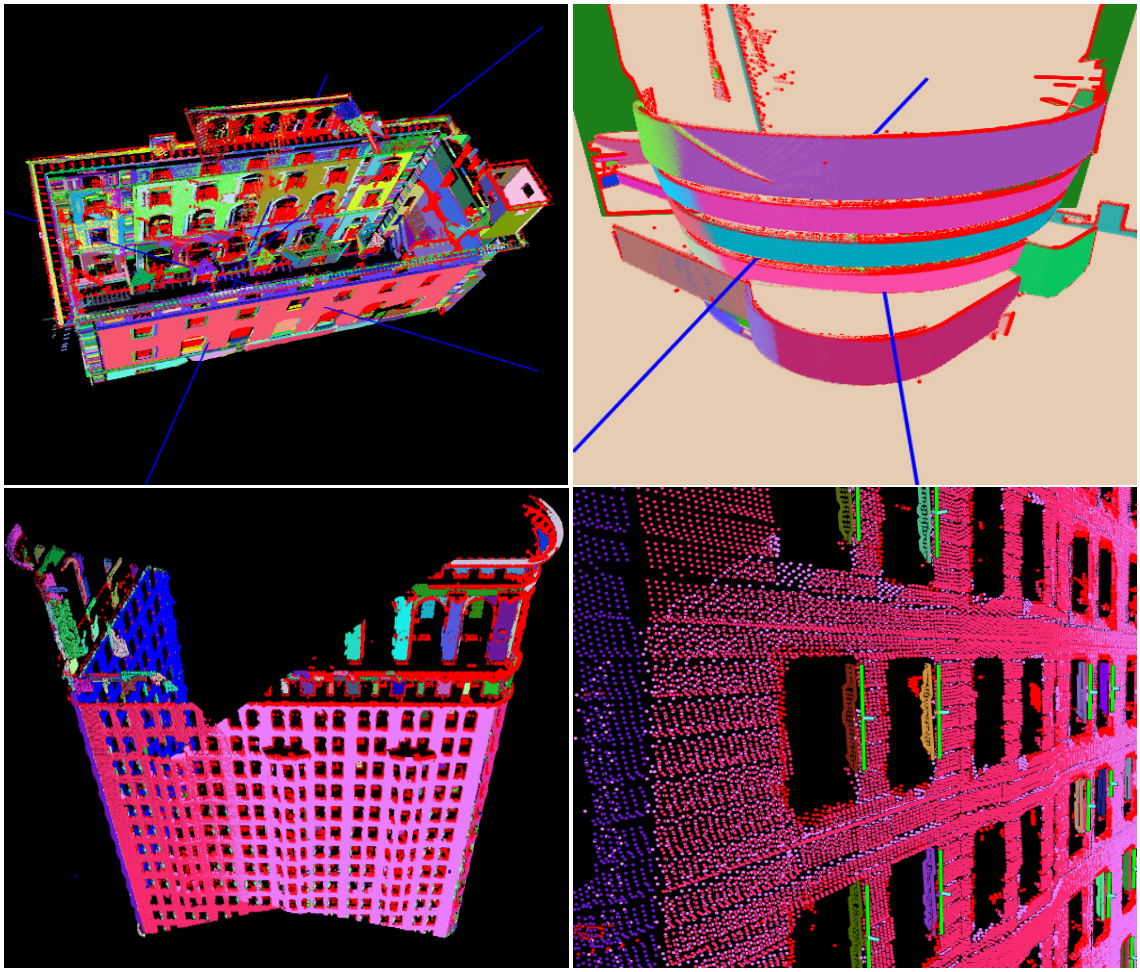

Figure 4.3: Registration of a) 3 range scans of the Casa Italiana, b) 2 range scans of the Guggenheim Museum and c) 2 range scans of the Flat Iron Building. d) Close view of registration of Flat Iron Building.

Results of the registration algorithm on three data sets are presented in figure 4.3.

4.3.2 Hole filling via linear interpolation

Another issue which we need to consider is the fact that parts of the model may be missing due to transparent objects in the scene (such as windows). Those parts

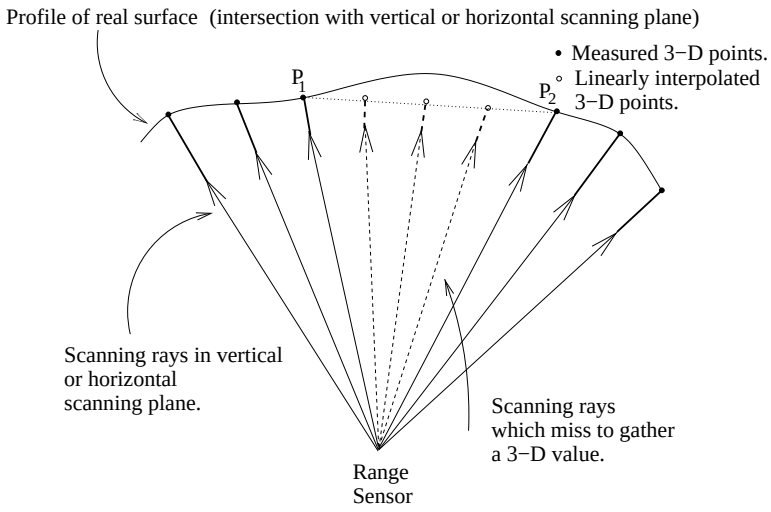

Figure 4.4: Linear interpolation of points along vertical or horizontal scan line.

need to be interpolated from neighboring samples. The first step is the computation of the average angle between vertical scanning planes (α) and between horizontal scanning planes (β). Then the knowledge of those angles allows us to interpolate in the vertical or horizontal scanning direction as shown in figure 4.4.

4.3.3 Extrusion of polygonal features

Our main contribution to 3-D modeling is the incorporation of segmentation as the necessary stage of early 3-D modeling. The generation and unification of every individual triangular prism in order to create the final solid model (see figure 4.2) can be avoided with the use of polygonal prisms in the areas where planar surfaces have been extracted. Instead of creating one triangular prism for each individual mesh element we can create prisms for the large planar polygonal areas extracted by the segmentation process. Parts of the scene which can not be segmented into planar regions are treated as triangular prisms swept towards the sensing direction.

The basic idea is illustrated in figures 4.5 and 4.6. In figure 4.5 an example

of a planar face (which is the result of the segmentation process) is shown. The face is defined by its outer and two inner boundaries. That face is being defined by its one outer and two inner boundaries (the inner boundaries represent holes). The polygonal planar face is surrounded by a number of triangular mesh elements, which represent parts of the scene where a segmentation was not possible. Figure 4.6 demonstrates the result of sweeping the outer polygonal, hole and triangular elements in the sensing direction. In the top row a presentation of the sweeping when no holes exist is shown. Polygonal and triangular prisms are unified in order to create a partial solid model. The extent of the sweep operation is based on determining and adequate far plane distance that will envelope the building's extent (for the Italian house experiment we swept each volume back by 120 meters). A complication arises in the presence of hole elements. This complication is demonstrated on the bottom row of figure 4.6. Holes inside large polygonal elements must be handled differently; the prisms created by sweeping the holes can be thought of as negative prisms. That means that a hole defines a void space. This is naturally implemented by the subtraction of the swept volume generated by the holes from the final partial solid model.

4.4 Results

This section presents our results in generating the solid model of an urban structure on Columbia University's campus by implementing the methods described in this chapter. The acquired range scans are segmented, registered and finally transformed to a non-redundant volumetric solid representation. The segmentation of each range scan is done by means of the segmentation algorithm described in chap-

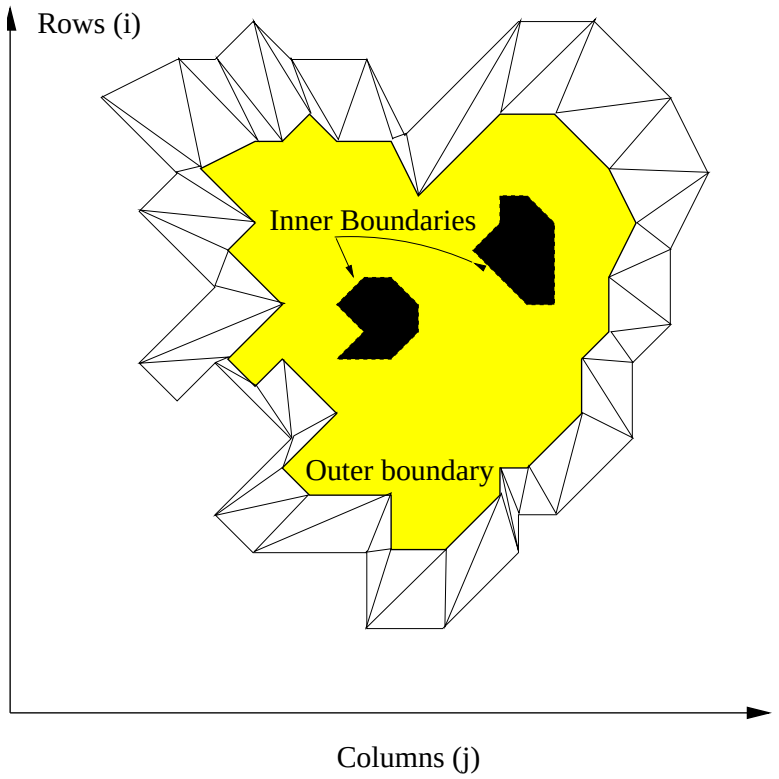

Figure 4.5: Modeling with segmented regions. Segmented region with outer and inner boundaries (see chapter 3). The boundaries of the region are shown on the regular grid over which the range image is defined. The polygonal boundaries of the planar regions are swept back in the direction of sensing. Parts of the scenes which can not be segmented (shown here to encompass the segmented polygonal face) are treated as triangular meshes which are swept back.

ter 3. Manual match of pairs of automatically extracted 3-D lines is used in order to align the scans on the same coordinate system (section 4.3.1). The results presented here describe the outcome of the solid model creation phase analyzed in sections 4.2 and 4.3.

The original range scans of the building Casa Italiana (figures 3.11, 3.12 and 3.13) are sub-sampled and then segmented into polygonal planar regions (table 4.1 contains the size of the sub-sampled meshes and table 4.2 the segmentation thresholds being used).

The unification of those two types of sweeps (sweeps produced by polygonal and sweeps produced by triangular faces) provides the final solid sweep for the first view of the building. The final solid sweeps for the three views of the building Casa Italiana and their boolean intersection which result in a topologically correct and geometrically accurate solid model of the scene are shown in figure 4.7.

Tables 4.3 and 4.4 present the reduction on the number of polygons and on the time the solid sweep algorithm takes to run. That reduction and increased efficiency is the result of the involvement of the segmentation process in the modeling phase. In table 4.3 the reduction in the number of triangular faces as a result of the segmentation is shown for the three range scans. The second column presents the number of triangular prisms that would be used for modeling if segmentation was not used. The third and fourth column show the number of triangular and polygonal prisms that are used for modeling after the segmentation of the range scans. On the last column you can see the amount of reduction on the number of triangular prisms due to the segmentation process. The reduction in the spatial complexity of the sweeps is very big (in the order of 50%). That reduced complexity greatly sim-

plifies the task of boolean intersection. It also increases the time-efficiency of the solid modeling phase. In table 4.2 the results of this improvement are presented. In the second column the time spent on segmentation is displayed. The third column is the time spent on the creation of the large polygonal prisms, whereas the forth column presents the time spent on the creation of the sweeps of the triangular sweeps (the algorithms did run on a Silicon Graphics Onyx2). What is eminent from this table is that modeling almost half of each scan (table 4.3) with polygonal prisms can be done orders of magnitude faster than the modeling of the unsegmented parts.

Mesh sizes	
Viewpoint	rows × columns
1	202 × 199
2	175 × 168
3	205 × 200

Table 4.1: Mesh sizes used for modeling.

Segmentation Parameters				
Viewpoint	$k \times k$	P_{thresh}	α_{thresh}	d_{thresh}
1	5 × 5	0.2	0.04^o	0.08m
2	5 × 5	0.2	0.04^o	0.08m
3	5 × 5	0.25	0.04^o	0.08m

Table 4.2: Parameters used for range segmentation.

Solid Modeling for Casa Italiana				
Viewpoint	Triang. prisms (before)	Triang. prisms (after)	Polyg. prisms	Reduction
1	79,596	41,864	32	52.6 %
2	58,116	24,480	19	42.1 %
3	81,192	41,245	34	50.8 %

Table 4.3: Results from modelers extension: reduction on number of triangular prisms.

Solid Modeling for Casa Italiana			
Viewpoint	Time (segmentation)	Time (polyg. prisms)	Time (triang. prisms)
1	13 secs	12 secs	152 mins 32 secs
2	16 secs	13 secs	77 mins 36 secs
3	11 secs	24 secs	184 mins 13 secs

Table 4.4: Results from modelers extension: time reduction.

4.5 Summary

Reed's system is based on the transformation of each input cloud of ordered range points into a triangular mesh and then in sweeping that mesh in the sensing direction in order to create a partial solid model of the scene. Merging of individual registered range scans is then possible by well-defined boolean CAD intersections (which are robustly implemented by commercially available CAD modelers) between the partial solid volumes those scans have been transformed into. Experiments performed using the original system provide excellent results for small objects scanned in the controlled laboratory environment. Large urban scenes, though, drive the system to its limits. The big number of samples per view increases the size of the input meshes and thus the time and space complexity of boolean CAD intersections. Our solution to this problem includes intertwining the solid-modeling part with the segmentation process.

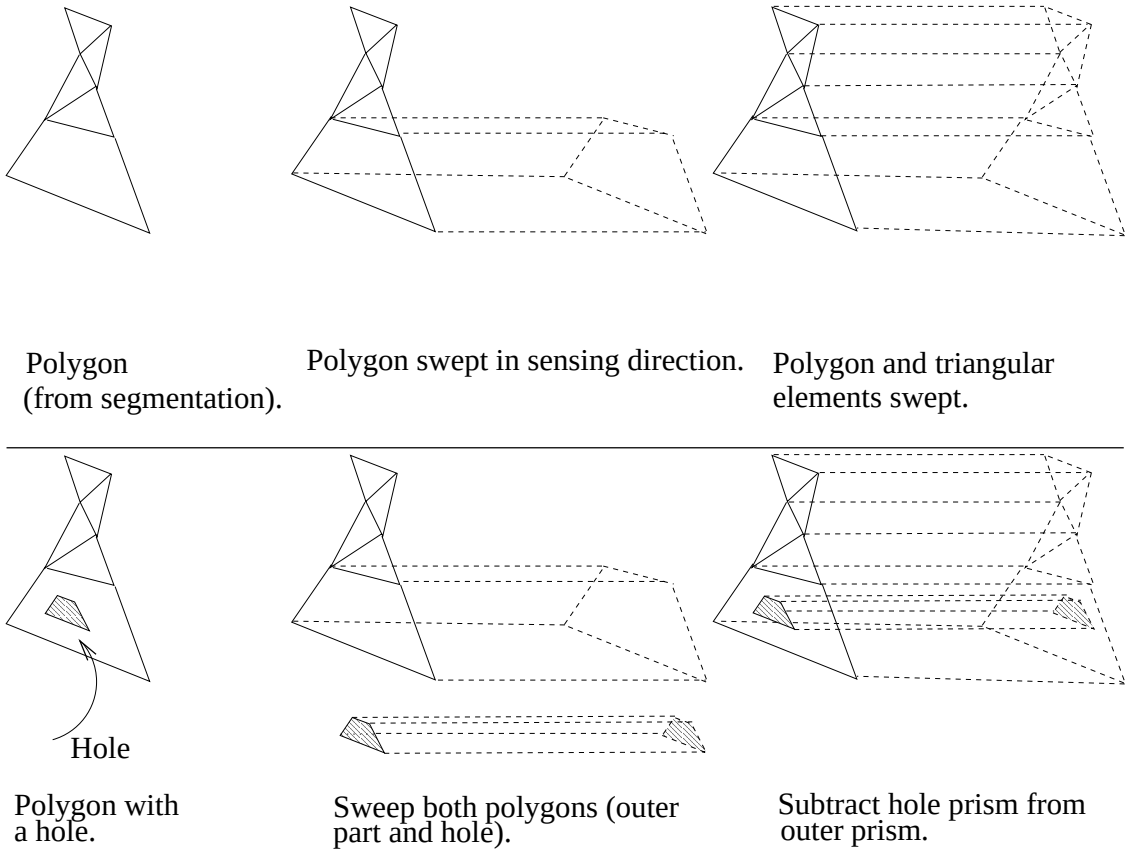

Figure 4.6: Extension of solid modeler. Top row: a)(left) Polygonal face (without holes) and triangular mesh elements, b) One polygonal element swept toward the sensing direction, c) All elements swept. The union of all prisms is the resulted partial solid volume. Bottom row: same concept. Now, though, the polygonal face has holes. a)(left) Polygonal face (with holes) and triangular mesh elements. b) One polygonal element swept toward the sensing direction. The hole is swept as well. c) All elements swept. The union of all prisms minus (boolean subtraction) the hole prisms is the resulted partial solid volume.

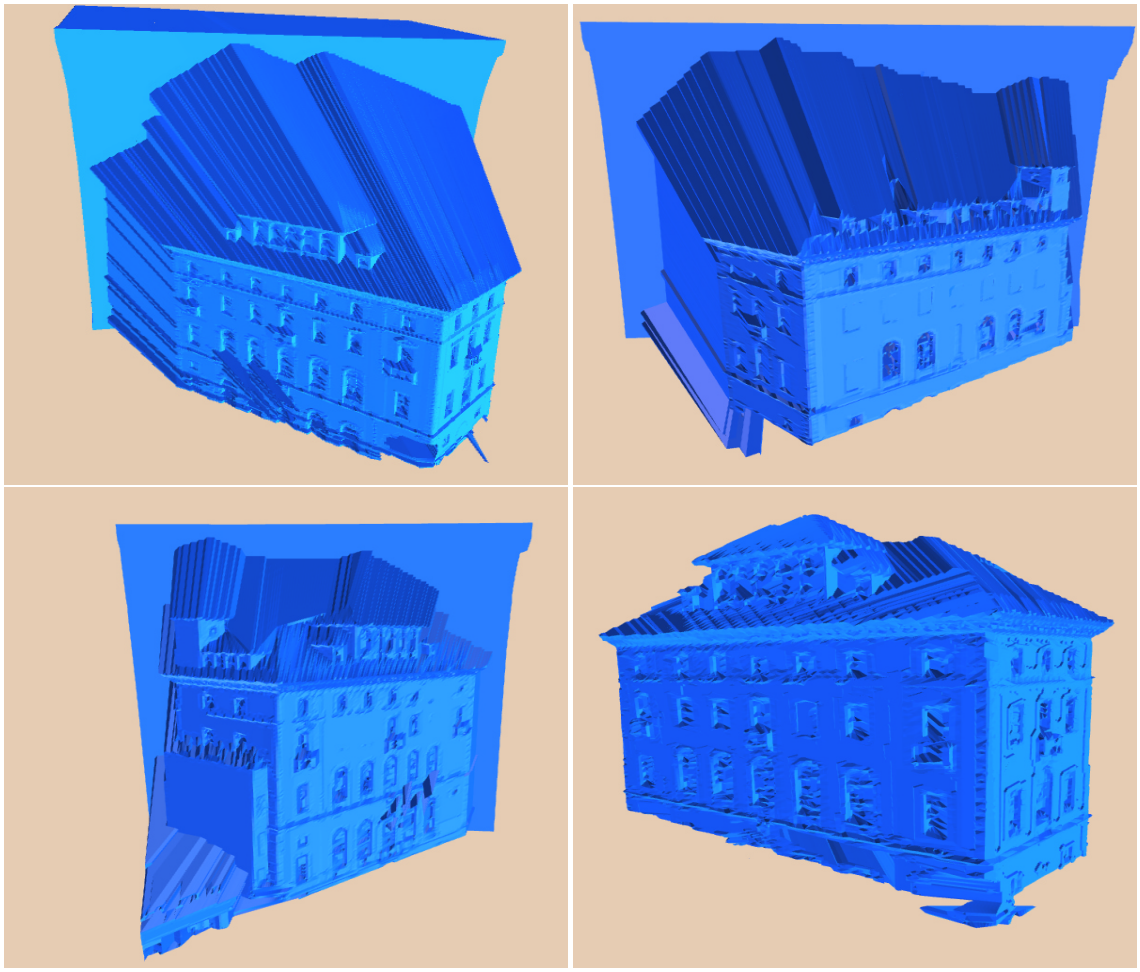

Figure 4.7: Modeling Casa Italiana. Three volumetric sweeps and final composite solid model.

Chapter 5

Range to Image Registration

In this chapter we deal with the problem of automatic pose estimation and calibration of a camera with respect to the acquired geometric model of an urban scene. The knowledge of the camera position, orientation and calibration parameters allow us to invert the image formation process and project the image of the camera back to the geometric model of the scene. In this sense we have a complete photo-realistic representation (dense geometry provided by the range sensor combined with a set of acquired 2–D images), a representation that can be used for efficient rendering. In the context of this thesis an automatic method for the registration of an image (from the 2–D camera) with a geometric model (from the 3–D sensor) is provided. This method is currently applicable only for urban scenes where regular patterns of rectangles (windows, doors) exist and can be sensed by both the range and image sensors.

As mentioned before, most systems which recreate photo-realistic models of the environment by a combination of range and image sensing [VIT, 2000, Pulli *et al.*, 1998, Zhao and Shibasaki, 1999, Sequiera *et al.*, 1999, McAllister *et al.*,

1999] solve the range to image registration problem by fixing the relative position and orientation of the camera with respect to the range sensor (that is the two sensors are rigidly attached on the same platform). The camera calibration is done off-line by sensing scenes of known geometry. In this case, the image to range registration problem is transformed to a range to range registration problem[1]. The major drawbacks of this approach are the following:

Lack of 2–D sensing flexibility Since a 2–D image is always attached to a range image, the acquisition of a new 2–D image requires the acquisition of a range image as well. But, range acquisition is much slower than image acquisition. Also there are limitations on where a range sensor can be placed with respect to the scene (standoff distance, maximum distance) which translate to constraints on the camera placement. As a result, the flexibility that the camera sensor provides (minimal constraints with respect to its position in the scene, fast acquisition and compact size) is being lost.

Static arrangement of sensors The system can not dynamically adjust to the requirements of each particular scene, since changes on the sensors relative configuration, or changes on the internal camera parameters (focal length for instance) require an off-line re-calibration before the scene acquisition.

Historical photographs The fixed methods do not have the ability of mapping historical photographs of buildings on the acquired geometric models[2].

In this thesis we provide a solution to the automated pose determination of

[1]The registration of the local coordinate frames of each range image with respect world is enough to align the camera images as well.

[2]Changes on the otter appearance of the building over time may result to further complications though.

a camera with respect to a range sensor without placing artificial objects in the scene and without a static arrangement of the range-camera system. This is done by solving the problem of automatically **matching** 3–D and 2–D features from the range and image data sets [Stamos and Allen, 2001].

5.1 Problem Formulation

Our goal is to provide an automated solution to the problem of registering 2–D camera images with 3–D range scans. We assume that both the camera and range sensors view the *same part* of the real scene, so that the 3–D and 2–D views have significant overlap (figure 5.1).

Formally, our input consists of the pair $(D(S), I(S))$ of a scene's S range scan D and an image I. The location of the camera which produces the image I is unknown and must be automatically recovered. Thus the output is the pose $P_i = \{R_i, \mathbf{T_i}|\mathbf{Pp_i}, f_i\}$ which describes (a) the transformation (rotation R_i and translation $\mathbf{T_i}$) from the range–sensor to each camera–sensor's coordinate system and (b) the mapping (internal camera parameters: principal point $\mathbf{Pp_i}$ and focal length f_i) from the 3–D camera frames of reference to the 2–D image frames of reference.

The pose estimation involves the following six stages.

1. Extraction of two feature sets F_{3D} and F_{2D} (3–D and 2–D linear segments from the range and image data-sets).

2. Grouping of the 3–D and 2–D feature sets into clusters of parallel 3–D lines L_{3D} and converging 2–D lines [3] L_{2D} (by exploiting **global** properties of the

[3] Those lines define vanishing points on the image space.

feature sets) [section 5.2].

3. Computation of an initial pose estimate $P_0 = \{R, \mathbf{0}|\mathbf{Pp}, f\}$ (rotation and internal camera parameters) by utilizing the directions defined by the sets of 3–D and 2–D clusters L_{3D} and L_{2D} [section 5.3].

4. Grouping of the 3–D and 2–D line segments into higher level structures of 3–D and 2–D rectangles R_{3D} and R_{2D} (by exploiting **local** properties of the feature sets). Extraction of 3–D and 2–D graphs G_{3D} and G_{2D} of rectangles (by exploiting the repetitive **pattern** of scene and image rectangles) [section 5.4][4].

5. Automatic selection of a matched set of rectangular features C^o and computation of a pose $P^o = \mathcal{A}(C^o|P_0)$ by running a pose estimator algorithm $\mathcal{A}$ on a number of samples over the set of all possible matches $\mathbf{C} = \mathcal{P}(R_{3D} \times R_{2D})$[5] (computation of a **coarse** pose estimate) [section 5.5].

6. Refinement $P^R = R(P^o, F_{3D}, F_{2D})$ of the estimated pose P^o by using all available information computed so far (computation of a **fine** pose estimate) [section 5.6].

The matching between 3–D and 2–D linear features is driven by the implicit assumption that matched 3–D and 2–D lines are the result of depth discontinuities in the 3–D scene. The following sections describe all steps in more detail.

[4]Those graphs are not utilized in the matching phase though.
[5]$\mathcal{P}(A)$ is the power set of a set A.

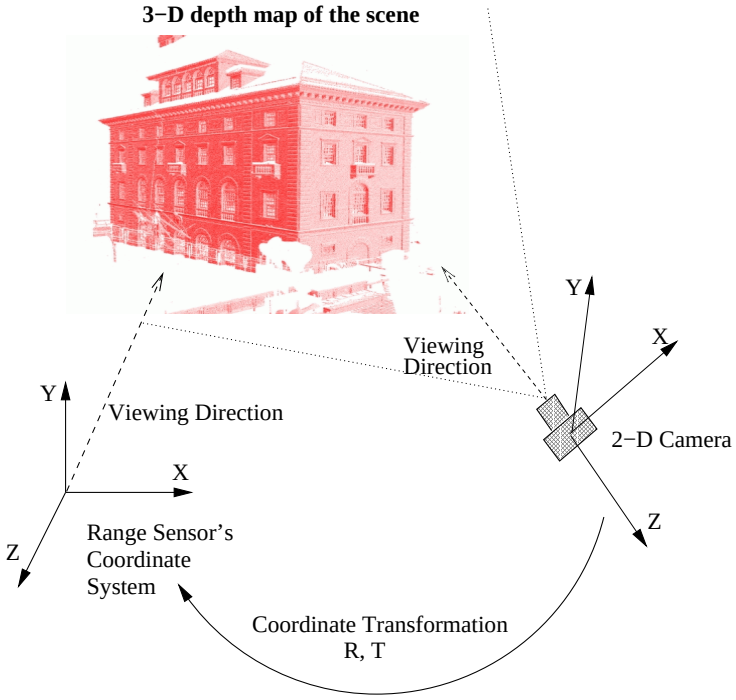

Figure 5.1: The pose estimation problem. The 3–D model of the scene is represented in the coordinate system of the range sensor. The image taken from the 2–D camera needs to be registered with the 3–D model.

5.2 Clustering 2–D and 3–D lines

Matched 2–D and 3–D clusters of lines are used for recovering rotation and for camera internal self-calibration. In the 2–D domain, the extraction of vanishing points provides a natural clustering of lines into sets which correspond to parallel 3–D lines whereas in the 3–D domain, the clustering into sets of parallel 3–D lines is direct. First we describe our vanishing point extraction algorithm and then the classification of 3–D lines. The end result is sets of 2–D lines [6] which meet at vanishing points and sets of 3–D parallel lines which produce the extracted vanishing points (figure 5.6).

[6]The canny edge detector with hysteresis thresholding [Canny, 1986] is used for extraction of 2-D edges. Those edges are grouped into linear segments via orthogonal regression.

Our belief is that vanishing points are a great source of information that maybe used in the context of urban scenes. The rotation that can be computed by matching scene directions with image vanishing points is a critical amount of information which can simplify the task of automatically computing the translation.

5.2.1 Vanishing Point Extraction

The most characteristic property of perspective projection is the fact that a set of parallel 3–D lines is mapped to a set of 2–D lines which intersect at a common point on the image plane. This point of intersection can be a point at infinity when the corresponding 3–D direction is parallel to the image plane. In order to handle all possible points of intersection (even points at infinity) we need to adopt the representation for 2–D points and 2–D lines described in appendix A. Then, the intersection of two 2–D lines $\mathbf{l_{12}}$ and $\mathbf{l'_{12}}$ is the point $\mathbf{v}$ which is mapped to the antipodal points $\pm\mathbf{N_{12}} \times \mathbf{N'_{12}}$ on the Gaussian sphere (figure 5.2) and can be represented by a pair (ϕ, θ).

There are many methods for the automatic computation of the major image vanishing points (a few references are [Antone and Teller, 2000b, Caprile and Torre, 1990, Wang and Tsai, 1990, Liebowitz and Zisserman, 1998]). Our approach involves the computation of all pairwise intersections between the extracted image lines and the creation of a 2–D histogram of intersections. The histogram is defined over the 2–D domain of the discretized surface of the Gaussian sphere. Then a search for the peaks of the histogram is performed. Each peak corresponds to directions towards which a large number of 2–D lines are converging.

Figures 5.3a–5.3c present the major points of our approach in more detail.

The representation of lines converging to the same vanishing point **v** lie on the same plane whose normal is **M**. The unit vector **M** is the representation of the vanishing point **v** (figure 5.3a). That means that by clustering the representations of all 2–D lines into major planes the goal of extracting the major image vanishing points is accomplished. The first step towards this goal is the consideration of all line–pairs P_i which define a local plane with normal M_i (figure 5.3b). Normals of line-pairs can be clustered into narrow areas as shown in figure 5.3b. Gaussian directions where a large concentration of such line–pairs is observed are the most probable candidates for the major scene vanishing points. So, the next natural step is to identify those places where intersections are being densely clustered. In order to accomplish that, we record all pairwise intersections on the discretized surface of the Gaussian sphere (figure 5.3c). The major clusters of intersection in the Gaussian sphere are then computed. The average direction of those clusters correspond to the representation **M** of the estimated vanishing points. Also all lines which are part of line-pairs contributing to each cluster are the ones that support each individual vanishing point.

The end result is a set of major vanishing points $\mathbf{VP} = \{VP_1, \ldots, VP_n\}$, where $VP_i = (\phi_i, \theta_i)$[7]. Each vanishing point is supported by a set of 2–D lines and the desired clustering $\mathbf{L_{2D}} = \{L_{2D_1}, \ldots, L_{2D_n}\}$ has been accomplished. If the number of major vanishing points N_{vps} is known a-priori (in urban environments this number is almost always three) then we can select the N_{vps} largest clusters from the set $\mathbf{L_{2D}}$ as our result. That is our final result is the set $\mathbf{L_{2D}} = \{L_{2D_1}, \ldots, L_{2D_{N_{vps}}}\}$ of N_{vps} 2–D line clusters and their corresponding vanishing

[7]The latitude-longitude representation depends on the assumed center of projection $\mathbf{C\tilde{O}P}$. See appendix A.

points $\mathbf{VP} = \{VP_1, \ldots, VP_{N_{vps}}\}$ (figure 5.4). Extracting the number N_{vps} is an easy task (it is equivalent to identifying the major modes of the 1–D histogram of directions of 2–D lines on the plane [Liebowitz and Zisserman, 1998]).

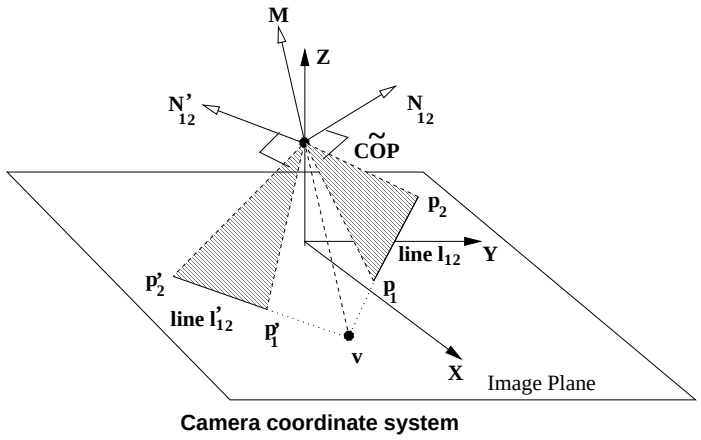

Figure 5.2: **Representation of point of intersection.** The point of intersection $\mathbf{v}$ of the two lines $\mathbf{l_{12}}$ and $\mathbf{l'_{12}}$ is represented by the pair $\pm\mathbf{M} = \pm\mathbf{N_{12}} \mathbf{x} \mathbf{N'_{12}}$ of antipodal points on the Gaussian sphere, where $\pm\mathbf{N_{12}}$ and $\pm\mathbf{N'_{12}}$ are the mappings of the lines $\mathbf{l_{12}}$ and $\mathbf{l'_{12}}$ on the sphere.

5.2.2 Clustering of 3–D lines

The clustering of the extracted 3–D lines into sets of parallel lines is an easier task than the extraction of vanishing points. We are using a classic un-supervised nearest neighbor clustering algorithm [Jain and Dubes, 1988].

Initially, each 3–D line l_i defines its own cluster C_i. Also a dissimilarity matrix $D(i,j)$ is initialized. The dissimilarity between 2 clusters C_i and C_j is equal to the angle between the average directions of lines belonging to those two clusters. After the initialization stage the algorithm finds the two closest clusters and merges them into one new cluster while it updates the dissimilarity matrix D. The algorithm repetitively finds and merges the two closest clusters until the

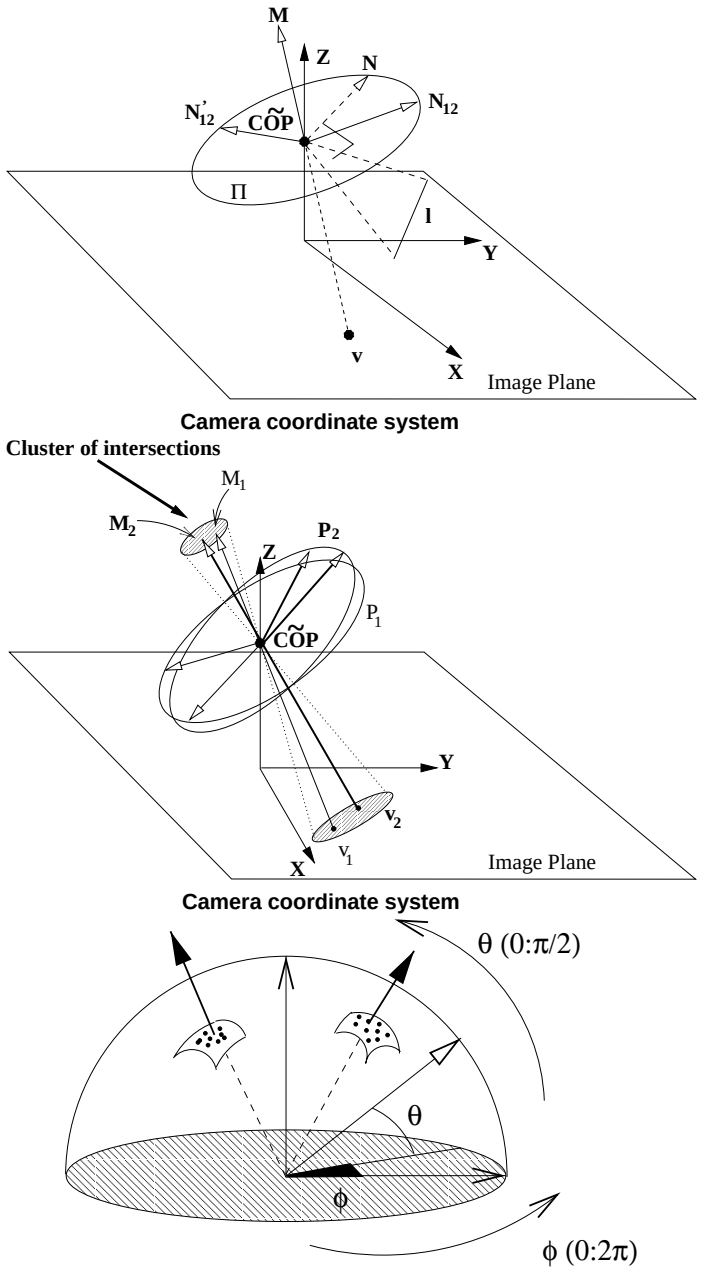

Figure 5.3: **a) (top) Vanishing point representation.** The representation **N** of all lines **l** which converge to the vanishing point **v** lie on a common plane whose normal is the vector **M**. Vector's **M** intersection with the image plane is the vanishing point **v**. **b) Clustering points of intersection.** Points of intersection of pairs of lines which converge to the same vanishing point are clustered in narrow areas (M_1 and M_2 are the 3–D vector representations of the intersection of pairs P_1 and P_2 whereas v_1 and v_2 are the actual image points of intersection). **c) Major clusters of intersections on the Gaussian sphere.** The major dense clusters of intersections are extracted by regularly sub-diving the surface of the sphere into patches.

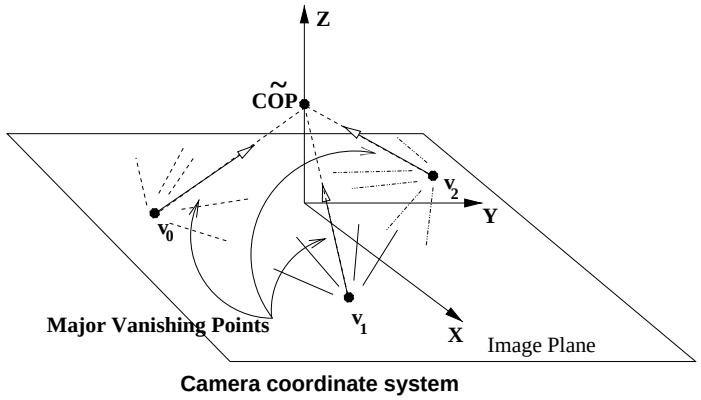

Figure 5.4: Extracted major vanishing points and their supported 2–D lines.

minimum distance between clusters is larger than a user specified maximum angle $\alpha_{3Dclust}$.

Finally, the N_{vps} (section 5.2.1) larger clusters of 3–D lines provide the desired grouping of 3–D lines into clusters of parallel lines $\mathbf{L_{3D}} = \{L_{3D_1}, \ldots, L_{3D_{N_{vps}}}\}$ along with the average 3–D direction of each cluster $\mathbf{U_{3D}} = \{u_{3D_1}, \ldots, u_{3D_{N_{vps}}}\}$.

5.3 Initial pose estimation

5.3.1 Solving for the rotation

The rotation computation is based on the fact that the relative orientation between two 3–D coordinate systems O and O' can be computed if two matching directions between the two systems are known (see figure 5.5). In this case there is a closed-form solution for the rotation ([Faugeras, 1996], page 523) and we can write $R = R(\mathbf{n_1}, \mathbf{n'_1} | \mathbf{n_2}, \mathbf{n'_2})$, where $\mathbf{n_i}$ and $\mathbf{n'_i}$ are corresponding orientations expressed in the coordinate systems O and O'.

In scenes containing a plethora of 3–D lines (such as scenes of urban struc-

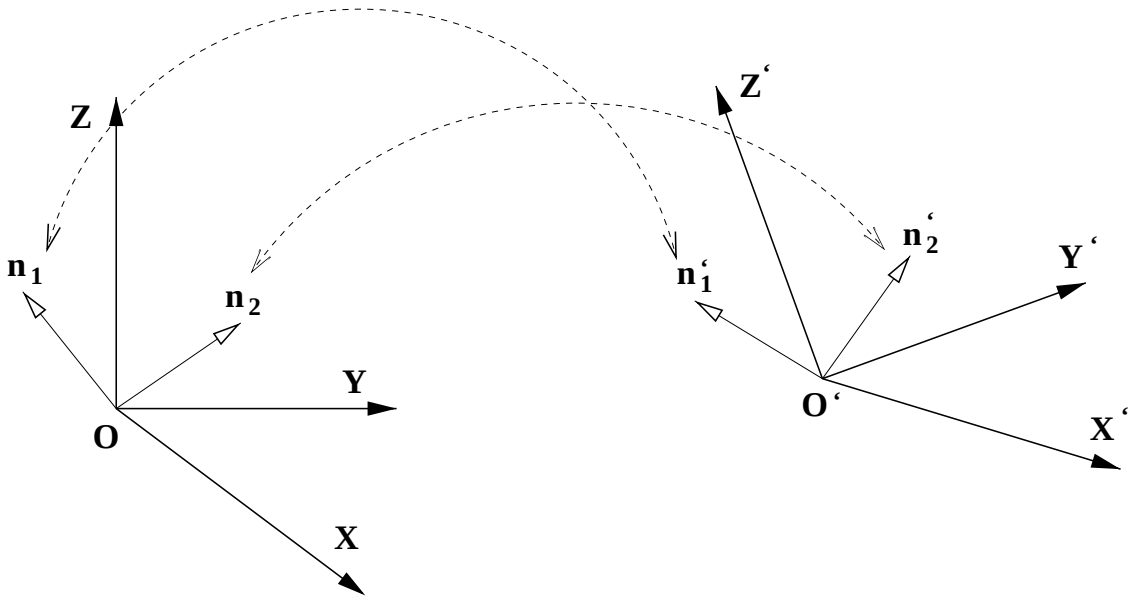

Figure 5.5: Matching directions between two coordinate systems

tures) it is possible to extract major 3–D directions with respect to the coordinate-systems of the range and image sensors. Those are the directions of clusters of parallel 3–D lines in the scenes (expressed in the coordinate system of the range sensor) and their corresponding vanishing point directions (expressed in the coordinate system of the image sensor) as shown in figure 5.6. We assume at this point that we have computed the camera center of projection (see section 5.3.2).

In more detail, the direction of the 3–D lines which produce the vanishing point v_i (figure 5.6) is the unit vector $\mathbf{n_i} = (v_i - COP)/||(v_i - COP)||$, expressed in the coordinate system of the camera sensor (appendix A). This direction can be matched with a scene direction $\mathbf{n_i'}$ which is expressed in the coordinate system of the range sensor and which has been provided by the 3–D clustering module (section 5.2.2). So, the rotation computation is reduced to the problem of finding

two pairs of matching 3–D directions and 2–D vanishing points

$$(\mathbf{n_i'}, \mathbf{n_i}) \, \epsilon \, \mathbf{U_{3D}} \times \mathbf{VP}.$$

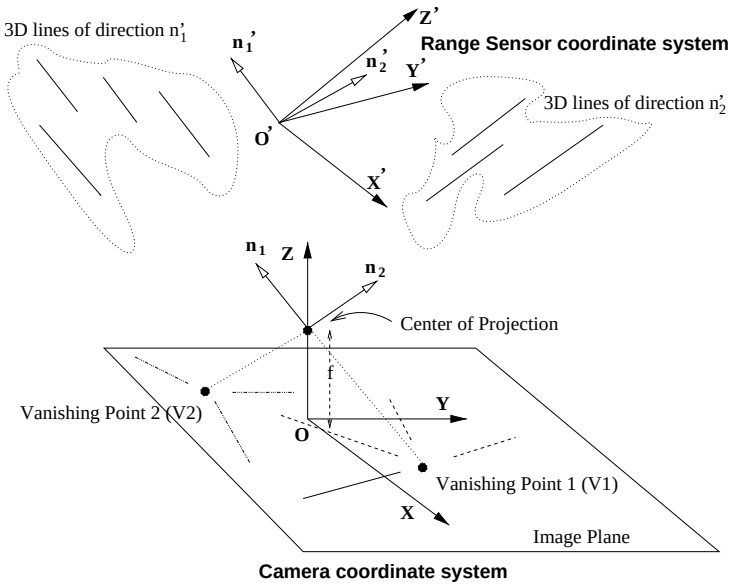

Figure 5.6: Two vanishing points. The 2–D lines which correspond to parallel 3–D lines of direction $\mathbf{n_i}$ intersect at a common vanishing point V_i on the image plane.

5.3.2 Camera Calibration

The extracted vanishing points can be used for the internal calibration of the camera sensor. We are using the technique which has been successfully used in urban environments by [Becker, 1997] for the computation of the focal and principal point of a 2-D camera by exploiting the following invariant property of parallelism:

- Parallel scene lines produce image lines which intersect at the same image point (vanishing points).

- Let three groups g_i of parallel scene lines of directions $\mathbf{L_i}$ produce the three vanishing points VP_i (figure 5.7). Then those vanishing points and the center

of projection of the camera form a proper tetrahedron if and only if the three scene line directions $\mathbf{L_i}$ are orthogonal wrt each other.

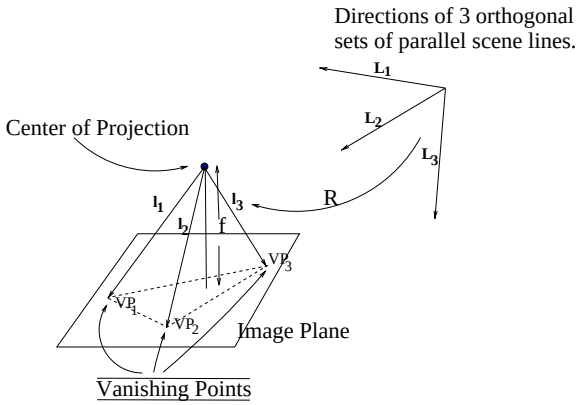

Figure 5.7: Calibration by utilizing three vanishing points.

5.4 Extracting 3–D rectangles and 2–D quadrangles

Recapping the previous section, the extraction of the global properties of the 3–D and 2–D data-sets (section 5.2) results in:

1. Clusters of 3–D lines $\mathbf{L_{3D}}$ and directions of those clusters $\mathbf{U_{3D}}$ (section 5.2.2).

2. Clusters of 2–D lines $\mathbf{L_{2D}}$ and their corresponding vanishing points **VP** (section 5.2.1).

Those global properties have been used for the calculation of the camera rotation and for the internal camera calibration as has been shown in the section 5.3. However, global properties alone are not enough for the translation calculation between the range and image sensor. Calculating the translation requires the exact matching

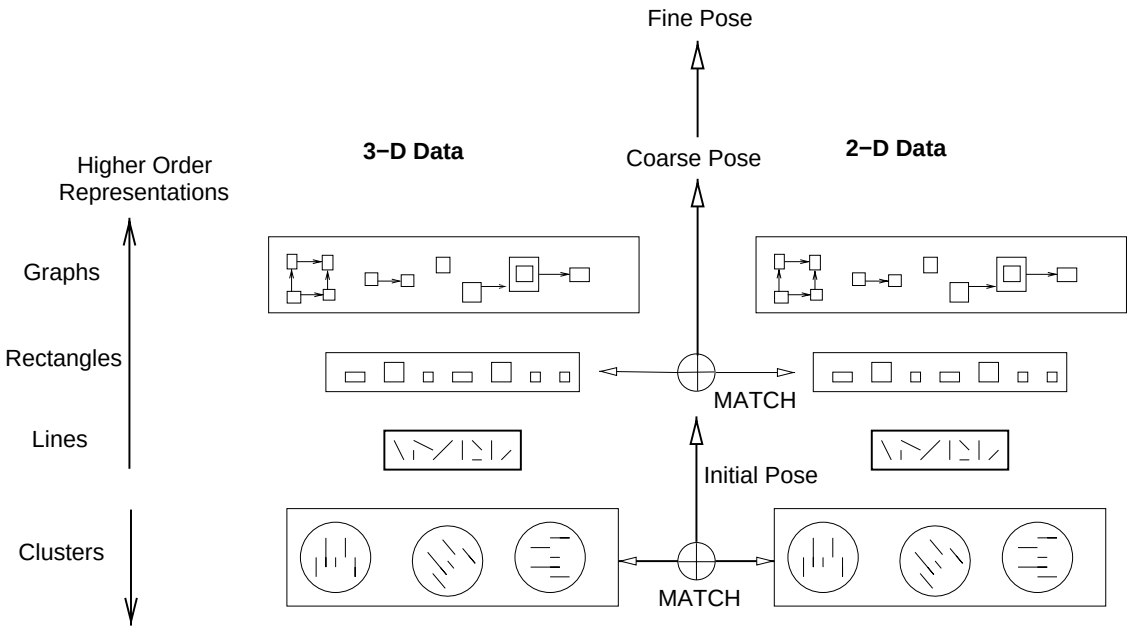

Figure 5.8: Hierarchy of 3–D and 2–D features and its use in the pose estimation algorithm. Matching of clusters of features provides an initial pose (rotation), whereas matching of individual rectangular structures provides a coarse pose estimate which is later refined.

of local 3–D and 2–D features (either points of lines). Since 3–D points are difficult to localize in the 3–D data set and since we have already developed a method for the reliable and accurate extraction of 3–D lines (section 3.5) we will match 2–D with 3–D linear features. Further, in order to reduce the search-space of possible matches we move up in the feature hierarchy presented in figure 5.8 and group the 3–D and 2–D lines into graphs of rectangular and quadrangular structures. Those rectangular and quadrangular structures will be used in order to determine which lines to match[8].

The geometry of the projection of a 3–D rectangle on a 2–D image quadrangle is shown in figure 5.9. 3–D rectangles which are formed by pairs of lines of directions

[8]The graphs can be potentially utilized in the matching process but they are not used in this thesis.

(V_{ver}, V_{hor}) have corresponding 2–D quadrangles which are formed by pairs of 2–D lines which converge to the vanishing points (v_{ver}, v_{hor}). That means that in order to extract corresponding 3–D rectangles and 2–D quadrangles we need to utilize the extracted clusters of 3–D and 2–D lines.

For the following discussion we will call one of the two scene directions **vertical** (V_{ver}) and the other one **horizontal** (V_{hor}). We assume that the vertical direction is oriented from the bottom to the top of the scene whereas the horizontal from left to right. Analogously we call v_{ver} and v_{hor} the vanishing points which correspond to the directions V_{ver} and V_{hor}.

We can formulate the 3–D and 2–D rectangle extraction problem as follows: The input is two pairs of 3–D directions $V_{ver}, V_{hor} \epsilon \mathbf{U_{3D}}$ and 2–D vanishing points $v_{ver}, v_{hor} \epsilon \mathbf{VP}$ along with the 3–D $L_{3D_0}, L_{3D_1} \epsilon \mathbf{L_{3D}}$ (section 5.2.2) and 2–D $L_{2D_0}, L_{2D_1} \epsilon \mathbf{L_{2D}}$ (section 5.2.1) clusters that support them. The output is a set of 3–D rectangles and 2–D quadrangles R_{3D} and R_{2D} and two corresponding graphs G_{3D} and G_{2D} describing the spatial relationship among structures in R_{3D} and R_{2D} respectively.

Following this notation a 3–D rectangle is a planar 3–D structure whose sides can be tagged as l_{top} or l_{bottom} if are parallel to the V_{hor} direction and as l_{left} or l_{right} if are parallel to the V_{ver} direction (figure 5.9). Also we can define three relationships between rectangles which lie on the same scene plane: *right of*, *top of* and *in* or *out of* (figure 5.11). Thus a 3–D rectangle can be viewed as the following tuple:

$$[Rec_{id}, Plane_{id}, \mathbf{size}, (l_{top}, l_{left}, l_{bottom}, l_{right}), (p_{top}, p_{left}, p_{bottom}, p_{right}), (p_{out}, \mathbf{p_{in}})]$$

where Rec_{id} is the rectangle's identification number,

$Plane_{id}$ is the rectangle's plane identification number,

size $= (s_{ver}, s_{hor})$ is the vertical and horizontal extent of the rectangle,

l_{dir} is the 3–D line defining the dir side of the rectangle,

p_{dir} is a pointer to the closest dir rectangle,

p_{out} is a pointer to the smallest rectangle enclosing Rec_{id} and

$\mathbf{p_{in}}$ is a set of pointers to all rectangles enclosed by Rec_{id}.

For the above notation we use the variable $dir \epsilon \{top, left, bottom, right\}$. The pointers p_{dir}, p_{out} and $\mathbf{p_{in}}$ can take the values Rec_{id} or $nill$ when there is no rectangle to point to.

The exact same representation can be used for the 2–D quadrangles [9]. In order to use the same notation and define spatial relationships between 2–D quadrangles we need to transform them to 2–D rectangles. This can be done if we rotate the two vanishing points v_{ver} and v_{hor} (and similarly transform all 2–D lines which they support them) such that they are parallel to the image plane (figure 5.10).

The rectangle-extraction problem is a search of patterns of 3–D lines and patterns of 2–D lines which have the structure shown in figure 5.9. After such patterns are detected the tuples defining each rectangle are being computed. The pointers p_{dir} describe the spatial relationship between rectangles and are thus describing the graphs G_{3D} and G_{2D}. Normally, our input consists of lines which do not define a complete four–sided rectangles. That is why we allow the representation and extraction of incomplete rectangles which are supported by less than four sides (in those cases on ore more of the entries l_{dir} are $nill$).

[9] The $Plane_{id}$ is not needed in this case since all 2–D structures lie on the image plane.

Our algorithms can detect false rectangles that do not exist in the real scene (since we extract rectangles supported by less than four sides). Our experiments show, though, that such an overestimation does not affect the outcome of the matching algorithm.

Following these guidelines the extraction of the 3–D and 2–D rectangles is a straightforward process. The next section provides the outline of the rectangle extraction algorithm.

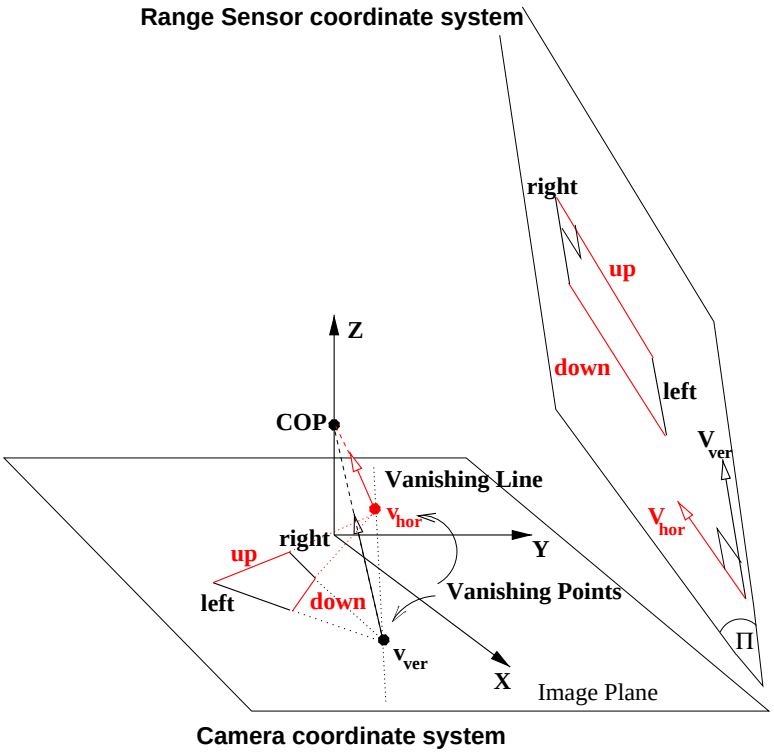

Figure 5.9: 3–D rectangle formed by lines parallel to the scene directions V_{ver} and V_{hor} and its corresponding 2–D quadrangle formed by 2–D lines which meet at the image vanishing points v_{ver} and v_{hor}.

Algorithm outline

Our rectangle extraction algorithm is almost identical for the 3–D and 2–D case.

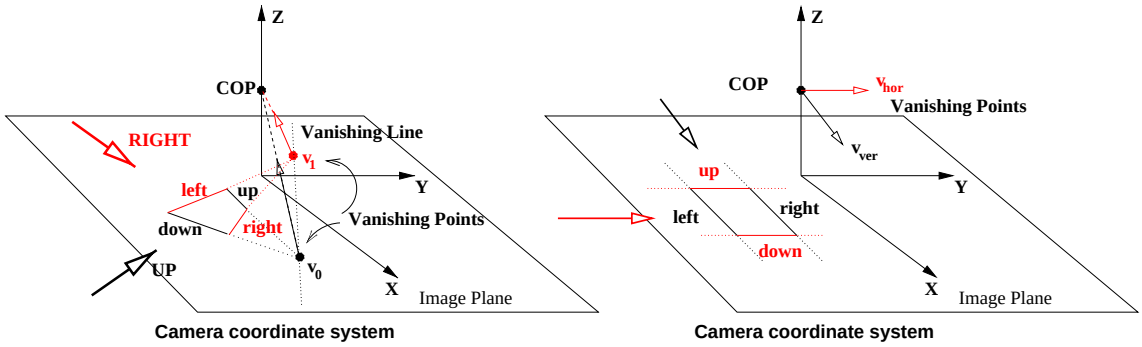

Figure 5.10: a)(Left) 2–D quadrangle under perspective projection, b) 2–D rectangle after the perspective effect has been canceled.

The differences between the 2–D and 3–D cases are the following:

2–D case: We need to extract quadrangles instead of rectangles (see figure 5.10a). However, with vanishing points already computed it is possible to undo the perspective effect and map quadrangles to rectangles (see figure 5.10b).

3–D case: We need to check for coplanarity of the linear segments that form the borders of the rectangle. Also, more information is available: the local planes that generate the 3–D edge segments are known.

In the discussion to follow we will present a general algorithm that can be applied in both 2–D and 3–D cases. The vertical and horizontal lines are directed according to the V_{ver} and V_{hor} directions. Thus each line can be represented as a pair of points (P_{start}, P_{end}). The major steps of the algorithm are the following:

1. Traverse all vertical lines (PV_{start}, PV_{end}) and record its four closest horizontal lines $(PH_{start_i}, PH_{end_i})$ in order to satisfy (if possible) the spatial relationships described in figure 5.12. Technically, you need to update the pointers connecting the endpoints of the vertical and its closest horizontal lines and reject all

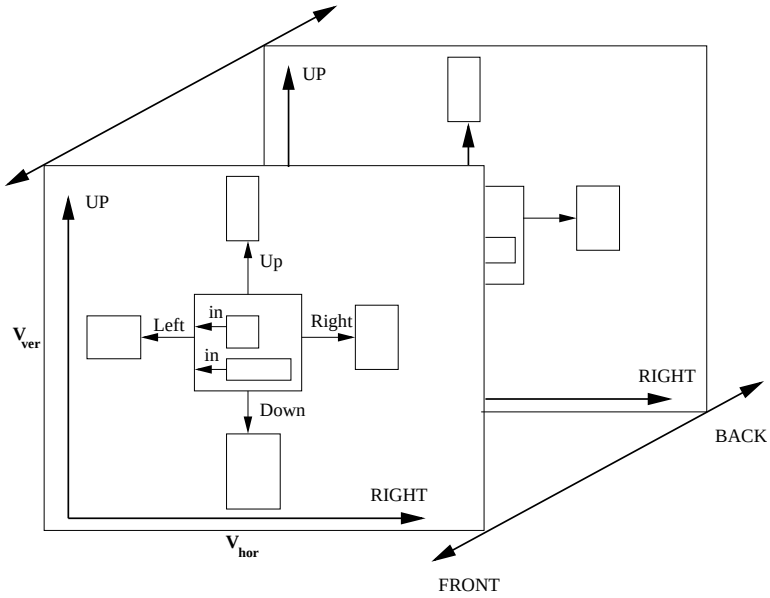

Figure 5.11: Spatial relationships between 3–D rectangles with sides parallel to the directions V_{hor} and V_{ver}. Rectangles lying on the same 3–D plane are related with the *Left*, *Up* or *In* relationships.

horizontal lines whose endpoints distance from the endpoints of the vertical line is more than a user supplied threshold $maxd$. The distance between a vertical and a horizontal line is defined as the distance between their closest endpoints.

2. Traverse the horizontal lines and check for patterns of four, three or two sided rectangles by utilizing the spatial relationships extracted in the previous step.

3. Compute the **size** (section 5.4) of all extracted rectangles[10].

4. Compute the graphs that describe the spatial relationships among rectangles (section 5.4).

Concluding, we have formulated and solved the problem of extracting 3–D

[10]Note the size of incomplete rectangles may be inaccurate due to lack of enough information.

and 2–D rectangles from pairs of 3–D directions $(V_{ver}, V_{hor}) \epsilon \mathbf{U_{3D}}$ (section 5.2.2) and their matching pairs of 2–D vanishing points $(v_{vert}, v_{hor}) \epsilon \mathbf{VP}$ (section 5.2.1). The output of this module is pairs of sets of 3–D and 2–D rectangles (R_{3D_I}, R_{2D_I}). In section 5.5 we will describe how we utilize the extracted sets of rectangles for the computation of a coarse pose estimate.

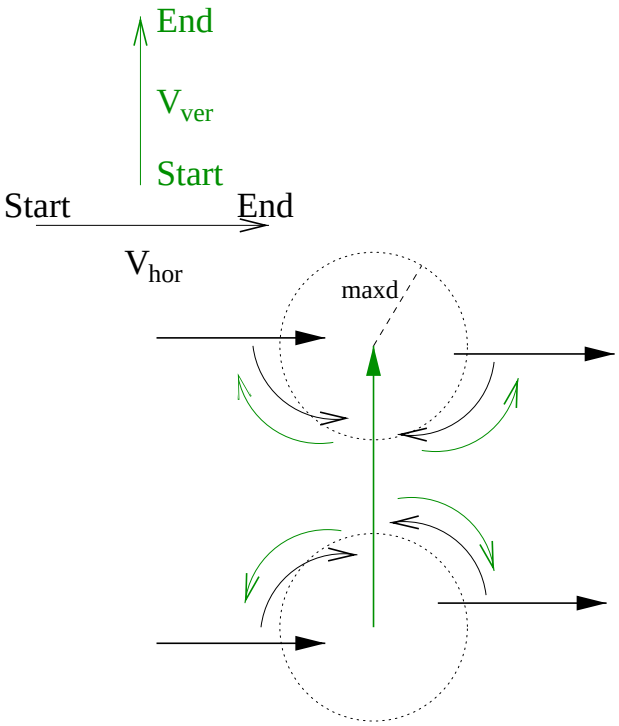

Figure 5.12: An oriented vertical line and its four closest horizontal lines. Pointers (relationships) between the end-points of the vertical and horizontal line are shown. The maximum distance between endpoints of vertical and horizontal lines is $maxd$.

5.5 Coarse Pose Estimation

So far, we have computed the rotation and internal camera calibration parameters of the camera by utilizing major vanishing points and 3–D directions in the scene. The last part of the pose computation module is the calculation of the camera

translation with respect to the range sensor by matching local 3–D and 2–D features between the range and image data sets. In section 5.3.1 3–D scene directions are matched with 2–D image vanishing points in order to solve for the camera rotation. Let N be the number of matched 3-D scene $\mathbf{n}'_\mathbf{i}$ and 2-D image $\mathbf{n}_\mathbf{i}$ directions and $M = \binom{N}{2}$ be the pairs of 3-D $(\mathbf{n}'_\mathbf{i}, \mathbf{n}'_\mathbf{j})$ directions which match with pairs of 2-D $(\mathbf{n}_\mathbf{i}, \mathbf{n}_\mathbf{j})$ directions. In section 5.4 we described a method to compute 3–D and 2–D rectangles (R_{3D_k}, R_{2D_k}) from clusters of 3–D and 2–D lines, and pairs of the form $((\mathbf{n}'_\mathbf{i}, \mathbf{n}'_\mathbf{j}), (\mathbf{n}_\mathbf{i}, \mathbf{n}_\mathbf{j}))$. In other words, 3-D range lines parallel to the two 3-D directions $(\mathbf{n}'_\mathbf{i}, \mathbf{n}'_\mathbf{j})$ produce a set of 3-D rectangles R_{3D_k} and 2-D image lines parallel to the two 2-D directions $(\mathbf{n}_\mathbf{i}, \mathbf{n}_\mathbf{j})$ produce a set of 2-D quadrangles R_{2D_k}. Some of the 3-D rectangles from R_{3D_k} may have potential matches on the 2-D quadrangles R_{2D_k}. That means that the set $R_{3D_k} \times R_{2D_k}$ contains pairs of potentially matched 3-D and 2-D structures. The following set

$$\mathcal{S} = \mathcal{P}(R_{3D_1} \times R_{2D_1}) \cup \mathcal{P}(R_{3D_2} \times R_{2D_2}) \cup \ldots \cup \mathcal{P}(R_{3D_M} \times R_{2D_M})^{11}$$

is the space of every possible matching configuration between 3–D and 2–D rectangles.

Exploring every possible combination of matches is an intractable problem since an exponentially large number of possibilities needs to be considered. In order to solve the problem we follow the RANSAC framework introduced in [Fischler and Bolles, 1981]. Instead of considering all possible matches we are randomly sampling the search space $\mathcal{S}$ by selecting sets C_{ransac} of n_{ransac} pairs of matched 3-D and 2-D structures (n_{ransac} is the minimum number of matches that can produce a reliable

[11] $\mathcal{P}(A)$ is the powerset of a set A.

pose-estimate).

RANSAC algorithm

1. Select at random a set C_{ransac} of n_{ransac} pairs of 3-D and 2-D structures.

2. Compute the pose $P_{random} = \mathcal{A}(C_{ransac}|P_0)$, where $P_0 = \{R, \mathbf{0}|\mathbf{Pp}, f\}$ is the initial pose computed in section 5.3, and $\mathcal{A}$ is a pose-estimator described in appendix B.

3. Project all rectangles of the set $\mathcal{R}_{3D} = R_{3D_1} \cup R_{3D_2} \cup \ldots \cup R_{3D_M}$ on the image assuming pose P_{random}.

4. Verify the existence of an acceptable match by comparing the projected 3–D rectangles $P_{random}(\mathcal{R}_{3D})$ with the set of extracted 2–D rectangles $\mathcal{R}_{2D} = R_{2D_1} \cup R_{2D_2} \cup \ldots \cup R_{2D_M}$ and compute a matching score Q_{match}.

5. Repeat the steps 1 – 4 N_{Max} times.

6. Finally, select as correct the match which produced the maximum matching score Q_{match}.

Maximum number of steps

According to [Fischler and Bolles, 1981] if we want to ensure with probability Pr that at least one of our random selections corresponds to a valid match then the maximum number of steps is

$$N_{max} = \log(1 - Pr) / \log(1 - b)$$

where b is the probability of randomly selecting a sample of n_{ransac} **correct** matches. If we assume that in our scene there are K pairs of 3–D and 2–D rectangles that can be correctly matched then

$$b = (K/L)^{n_{ransac}}$$

and

$$L = |(R_{3D_1} \times R_{2D_1}) \cup (R_{3D_2} \times R_{2D_2}) \cup \ldots \cup (R_{3D_M} \times R_{2D_M})| = n_1 m_1 + \ldots + n_M m_M$$

is the number of all possible pairs of 3–D and 2–D rectangles (where $n_i = |R_{3D_i}|$ and $m_i = |R_{2D_i}|$). Note that the lower the probability of correct matches b the larger the number of required steps N_{max}. Section 5.7 provides the exact number of steps used in our experiments [12].

Verification

In step 4 of the RANSAC algorithm the set of projected 3–D rectangles is being compared to the set of extracted 2–D quadrangles. The better aligned are the sets of 3–D and 2–D structures the larger the score Q_{match} of the match. Our algorithm sets Q_{match} to equal the number of 3–D rectangles which map (when projected to the image) to an extracted 2–D quadrangle.

What remains to be defined is how do we decide when two 2–D rectangles are close with respect to each other. Consider the two 2–D rectangles whose sides are oriented according to two directions we consider to be vertical and horizontal (figure 5.13) and let the sizes of the rectangles R and R' be (S_{ver}, S_{hor}) and

[12] One issue is the fact that the parameter K is unknown to us. We can assume that K is a large fraction of L (L is the number of all possibly matched rectangles). For our experiments we set $K = 1/3L$.

(S'_{ver}, S'_{hor}) (the sizes are part of the rectangles representation, see section 5.4). The two rectangles are close with respect to each other if and only if the distances between their corresponding sides is less than a threshold which depends on the size of both rectangles. This threshold is $\min(a_{ver}S_{ver}, a_{ver}S'_{ver})$ for vertical edges and $\min(b_{hor}S_{hor}, b_{hor}S'_{hor})$ for horizontal edges, where $a_{ver}, b_{hor} \epsilon (0, 0.5]$. The parameters a_{ver}, b_{hor} are supplied by the user.

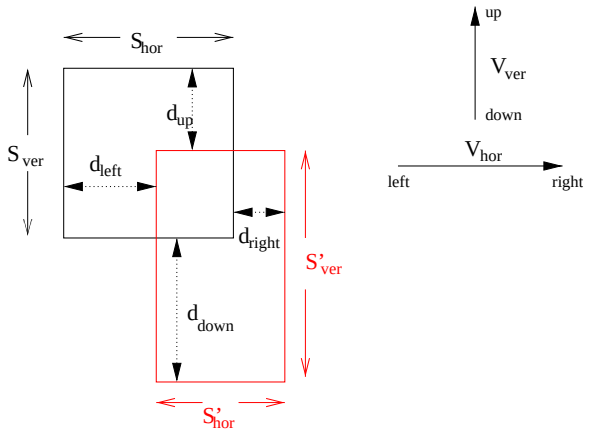

Figure 5.13: Defining the distance between two rectangles.

5.6 Final Pose Estimation

The coarse estimate computed in the previous section is very important because it provides an initial solution which can be subsequently refined. The refinement involves the projection of all 3–D lines of the extracted clusters $\mathbf{L_{3D}}$ on the 2–D image assuming the coarse pose estimate P^o and so a set of projected 3–D lines $P^o(\mathbf{L_{3D}})$ is formed. Each individual projected cluster is compared with the groups of extracted 2–D lines $\mathbf{L_{2D}}$ and new line matches among the 3–D and 2–D data sets are verified. A projected 3–D line matches the closest 2–D line on the image plane,

if the distance from the closest line is smaller than a user-specified threshold f_{thresh}. The distance between a projected 3–D and a detected 2–D line is defined as the minimum Euclidean distance between their respective endpoints [13]. The increased number of line matches results in better pose estimation. The pose-estimation algorithm used for the calculation of the final rotation and translation is described in appendix B. It is our implementation of the method described [Kumar and Hanson, 1994].

5.7 Results

In this section we present results for the automatic pose estimation between range and image data for an urban building. The results of the 3-D line and rectangle extraction from two range scans of Casa Italiana (figures 3.11 and 3.12) are shown in figures 5.14a and 5.14b. For clarity different rectangles are rendered with different color. Also there are three major clusters of parallel lines (encoded with the colors red, green and blue).

In the 2-D domain, the three major vanishing points and clusters of 2–D lines are shown in figures 5.15a and 5.15b. The automatically computed principal point of the cameras is also shown; it is the point of intersection of vanishing point directions on the image.

The next set of figures (5.15c,5.15d) display the results of extracting 2-D quadrangles from the 2-D images. The extracted quadrangles from two different views are overlayed on the 2-D images. Notice that our algorithm may extract

[13] Note that the projected 3–D and detected 2–D lines are parallel on the image plane since the rotation has already been computed.

"fake" quadrangles due to the fact that it is hard to extract all four borders of them. That means that we rely on incomplete information and that we have to introduce "fake" borders (that is lines which have not been extracted by our feature detector) in order to compute complete four-sided quadrangles.

The 2-D quadrangle extraction algorithm is described in more detail in figures 5.17 (first view) and 5.18 (second view). Figure 5.17a represents two clusters of 2-D lines which belong to two vanishing points (red: vertical, green: horizontal) after rectification (see figure 5.10). The same results is presented in figure 5.17b, but now a different pair of vanishing points is used (red: vertical, green: horizontal). The result of the 2-D quadrangle extraction in the rectified space is shown in figures 5.17c and 5.17d. It is possible to define graph structures in that space. Also the distance between a 2-D and a projected 3-D rectangle is defined in the rectified space (figure 5.13). Figure 5.18 presents the same set of results for the second view of the building.

The outcome of the coarse pose estimation algorithm (RANSAC algorithm) is presented next in figures 5.15e and 5.15f. The extracted 2-D rectangles (red) are shown overlaid with the projection (green) of those 3-D rectangles which produce the maximum matching score Q_{match} (Q_{match} is 9 for the first view and 8 for the second view). The final pose (section 5.6) is visually verified in figures 5.16a and 5.16b where the extracted 3-D lines shown in figures 5.14a and 5.14b respectively are projected on the 2-D images assuming the final pose. The extracted 2-D lines are shown in red and the projected 3-D lines in green. As you can see the projected 3-D lines are very well aligned with the 2-D data-sets, which means that both the registration and the feature extraction algorithms produce accurate results. The

number of samples the RANSAC algorithm tried was 8457 (6.0 seconds on an Onyx2) for the first view and 223831 (2 minutes and 29 seconds) for the second view.

Finally, the images 5.16c and 5.16d present the texture-mapped 3-D models using the computed calibration parameters and pose estimate on the two views of the model. The texture map, also visually verifies the accuracy of our method. The final pose estimates are $\mathbf{T} = (3.71, -2.93, 12.29)^T$ (in meters),
$R = \{175.65^o, (0.017, 0.99, 0.01)^T\}$ (angle-axis representation) for the first view and $\mathbf{T} = (1.35, -2.5, 10.10)^T$, $R = \{178.86^o, (0.0, 0.99, 0.01)^T\}$ for the second.

5.8 Threshold sensitivity

There are three types of thresholds used in this chapter.

- A threshold $maxd$ used in the rectangle extraction process (see section 5.4). In the 3-D case the value of $maxd$ was 0.5 m (for the first view of Casa Italiana) and 0.9 m (for the second view). In the 2-D case $maxd$ was 15 pixels for both views. Small values for $maxd$ result to under-estimation of the number of extracted rectangles, whereas large values lead to the extraction of fictitious rectangles. Large values can be used since fictitious rectangles are filtered out in the matching process.

- Thresholds used for matching (N_{Max}, a_{ver} and b_{hor}). The value of the maximum number of iterations of the RANSAC procedure N_{Max} is important, since an inadequate number of iterations would lead to a failure to compute the correct match between 3-D and 2-D features. The value of N_{Max} is con-

servatively estimated by setting a small value to the probability of a correct match[14] (see section 5.5). The thresholds a_{ver} and b_{hor} (see the last part of section 5.5) are used for the verification of overlap between a 2-D and 3-D projected rectangle. The smaller the values of a_{hor} and b_{ver} the larger the similarity between rectangles needed for verification. The algorithm is sensitive in the selection of those thresholds. In our experiments we used the maximum value of 0.5 units for both thresholds.

- A threshold f_{thresh} used in the final pose estimation routine of section 5.6 (set to the value of 50 pixels in our experiments). A large value for this threshold results in the detection of false matches between 3–D and 2–D lines and the fine pose estimate is worse than the coarse pose estimate. A small value on the other hand results in a few new matches and the coarse pose estimate does not improve significantly.

5.9 Summary

We have developed a method to accurately register a range with an image data set in urban environments. We are exploiting the parallelism and orthogonality constraints that naturally exist in such environments in order to match extracted sets of rectangular structures. The use of a RANSAC technique for the computation of an optimal match between the data-sets is feasible due to the reduction of the search space from the set of 3–D and 2–D lines to the set of 3–D and 2–D rectangles. The RANSAC technique on the other hand is the computational bottleneck since

[14]This probability is a-priori uknown to us though, and this small value is just a guess.

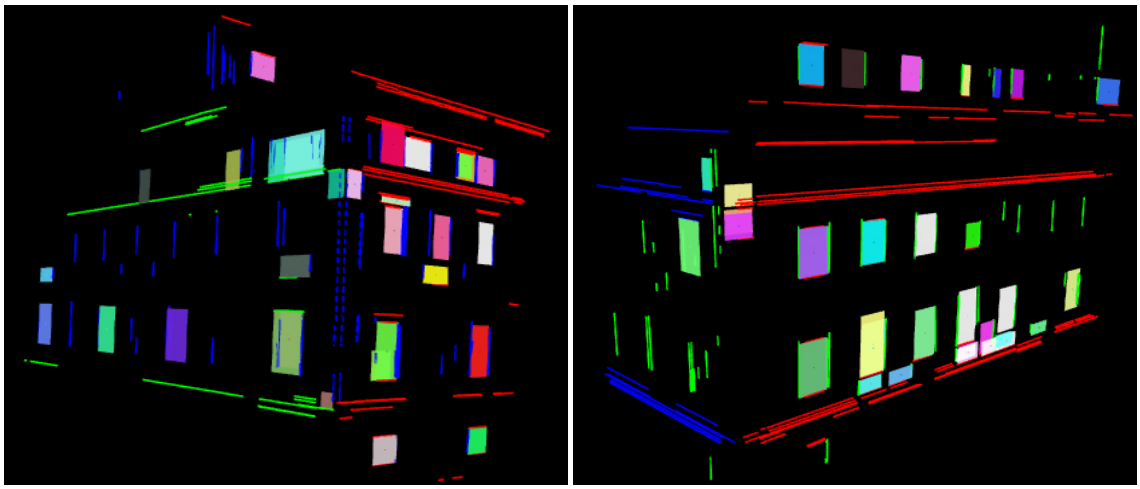

Figure 5.14: a,b) Clusters of 3–D lines (color encodes different directions) and extracted 3–D rectangles (rectangles are rendered as solids of different color for clarity). Two different views of the building.

a large number of potential matches need to be checked.

Figure 5.15: **Results.** a,b) 2–D images and clusters of 2–D lines, where different colors correspond to different vanishing points. c,d) Extracted 2–D quadrangles (two views). e,f) Extracted 2–D quadrangles (shown in red) and Q_{match} matched 3–D rectangles projected on images after coarse pose estimation (shown in green).

Figure 5.16: **Results.** a,b) Projected 3–D lines on the images after final pose estimation (shown in green). The extracted 2–D lines are shown in red. c,d) Images texture-mapped on 3–D model assuming final pose.

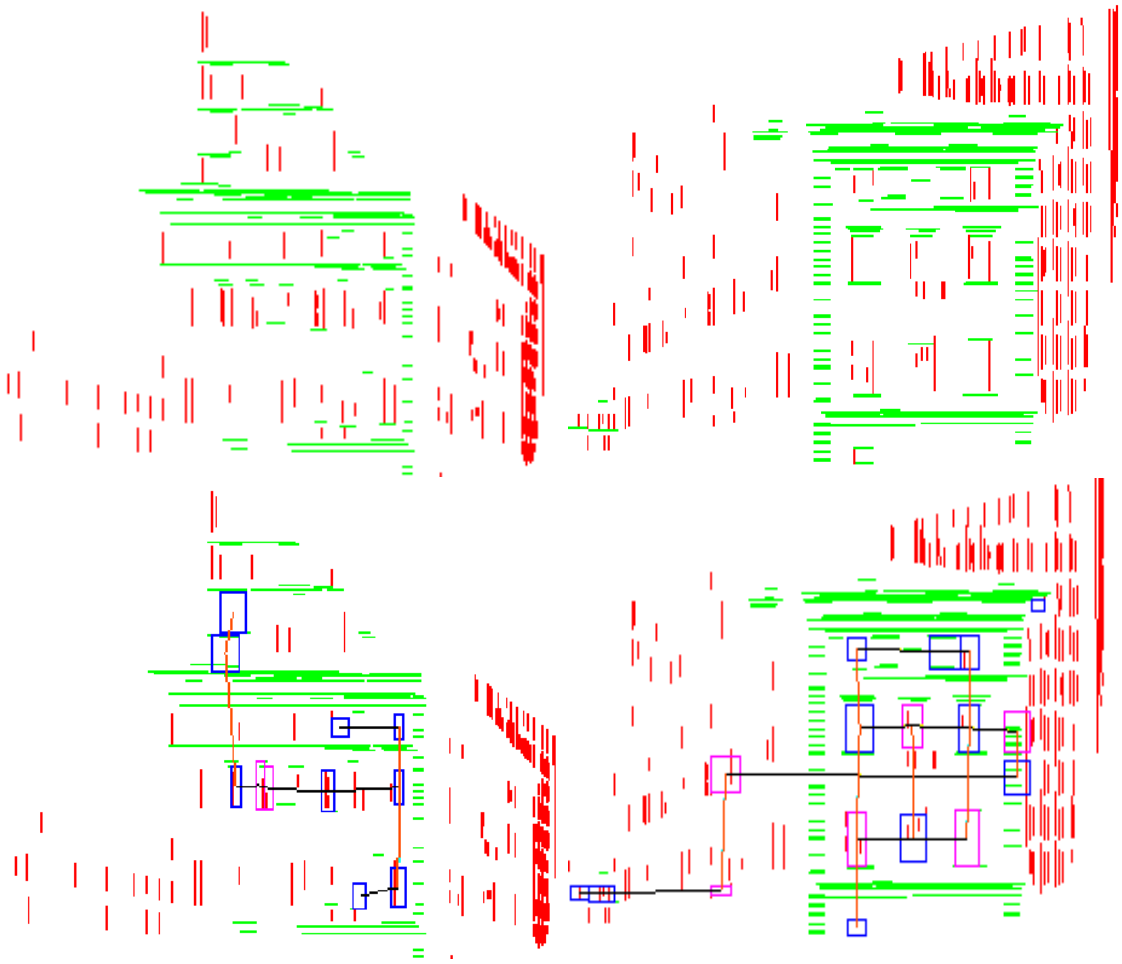

Figure 5.17: **2-D lines and 2-D graphs (first view).** Top row: a) Rectified 2-D lines corresponding to two vanishing points (red vertical, green horizontal towards the left of the scene). b) Rectified 2-D lines corresponding to two vanishing points (red vertical, green horizontal towards the right of the scene). Bottom row: c) Rectified 2-D rectangles (with graph relations) corresponding to rectified lines of (a). d) Rectified 2-D rectangles (with graph relations) corresponding to rectified lines of (b).

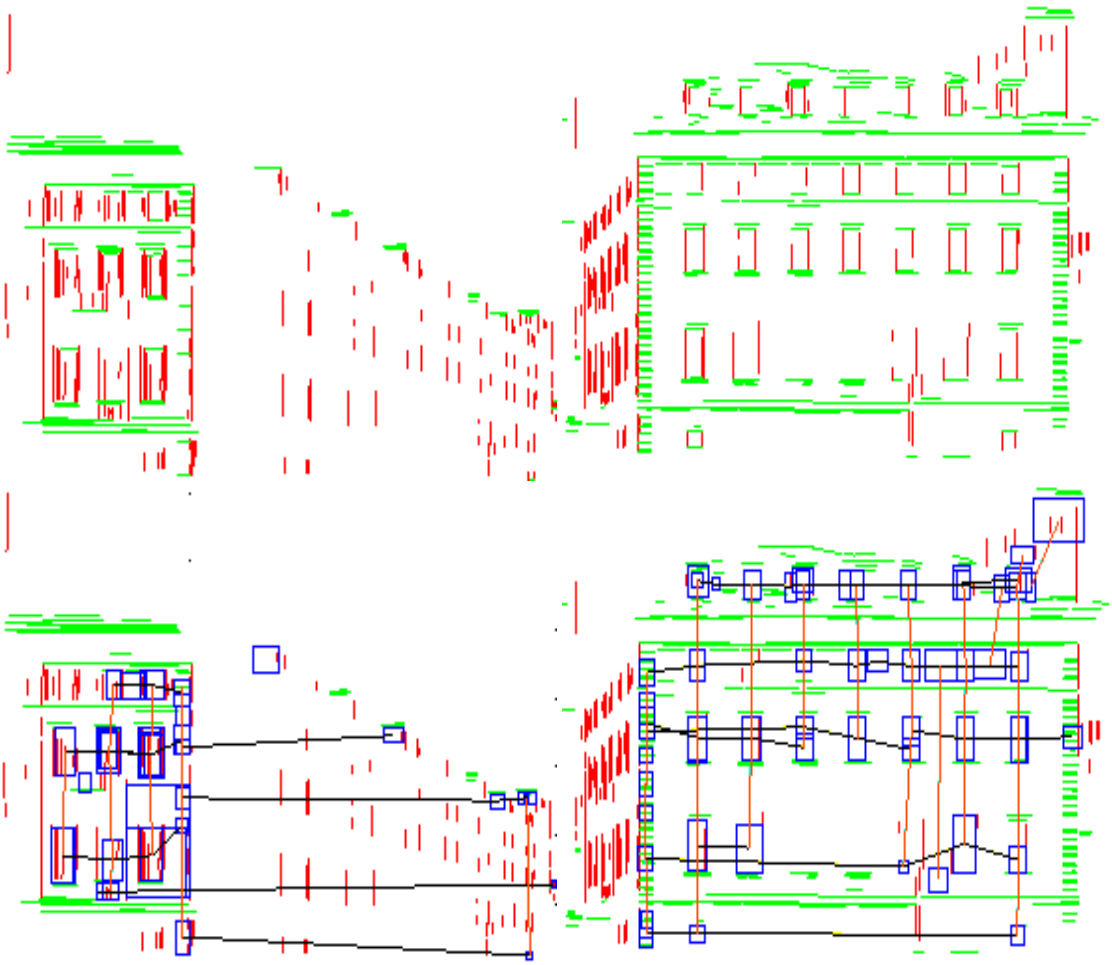

Figure 5.18: **2-D lines and 2-D graphs (second view).** Top row: a) Rectified 2-D lines corresponding to two vanishing points (red vertical, green horizontal towards the left of the scene). b) Rectified 2-D lines corresponding to two vanishing points (red vertical, green horizontal towards the right of the scene). Bottom row: c) Rectified 2-D rectangles (with graph relations) corresponding to rectified lines of (a). d) Rectified 2-D rectangles (with graph relations) corresponding to rectified lines of (b).

Chapter 6

Vision of the future

In this thesis, we address the need for highly realistic and accurate representations of the 3-D world. We designed and developed a system to recover geometric and photometric 3-D models by utilizing state-of-the-art range sensing technology and by enhancing the recovered geometric representation with photometric observations gathered by a conventional camera. The goal is the development of a comprehensive system, which can explore scenes of large scale, with a minimum of human intervention.

We feel that we have attacked one of the most difficult problems in computer vision and robotics research in a unique and effective manner. Below we summarize our contributions:

- Range segmentation and feature extraction algorithm (chapter 3). These methods allow us to drastically simplify the dense range data-sets and to extract high level features which can be used for registration purposes.

- Creation of geometrically correct solid models (chapter 4). Partial solid

sweeps are created by extruding the segmented planar surfaces along the sensing directions. The final model is the result of boolean intersection between the partial solid volumes.

- Automated range to image registration algorithm (chapter 5). In order to achieve this result we implemented various feature extraction algorithms (vanishing points, 3-D rectangles and 2-D quadrangles). The vanishing points are used for the calibration of the camera sensor and for the computation of the rotation between the camera and the range sensor. Finally, a method for the automated matching between 3-D and 2-D features provides a solution for the translation.

We believe that all these modules are of vital importance for a flexible photo-realistic 3-D model acquisition system. Segmentation algorithms simplify the dense data-sets and provide stable features of interest which can be used for registration purposes. The solid modeling provides geometrically correct 3-D models. The automated range to image registration can increase the flexibility of the system by decoupling the slow geometry recovery process from the image acquisition process; the camera does not have to be pre-calibrated and rigidly attached to the range sensor. Finally, sensor planning can be used for enhancing incomplete scene models by carefully choosing the position of the sensor in order to cover previously unseen parts (due to self-occlusions). The first steps in the direction of automatic sensor planning for outdoor scenes are described in section 6.3 of this chapter. This thesis presents the first photometrically enhanced solid CAD model of an urban structure created from range and image measurements.

6.1 Limitations

Due to the scope of the system, there are still a number of open technical issues that need to be addressed:

Segmentation The segmentation routines fit planes to the extracted clusters of points. Fitting of general smoothly varying surfaces is needed in non-planar parts of the scene. Also a number of thresholds have to manually be set by the user in order to customize the segmentation.

Range-Range Registration Our method is based on matching automatically extracted features between views. We did not automate the matching between 3-D features. We believe, however, that a RANSAC-type approach of matching 3-D lines or 3-D rectangular features can provide the desired registration results.

Solid Modeling The solid modeling part requires very accurate calibration between the range images. Also the whole object must be in the field of view of the sensor at each scanning operation. This problem can be attacked if we replace the boolean intersection of the solid sweeps with the unification of the complements of the sweeps.

Range-Image Registration Our algorithm operates in scenes which contain linear features with strong orthogonality constraints. We are not certain how this algorithm is going to extend to general 3-D scenes. Also in the self-calibration of the camera sensor the distortion is considered negligible.

Texture-mapping Currently we do not merge multiple images (textures) on the

3-D model but we texture-map only one brightness image per view. There are methods to attack the problem (i.e. view-dependent texturing [Debevec *et al.*, 1996] or statistical texture estimation [Coorg and Teller, 1999]). All approaches implement heuristics though and the research problem is still open.

6.2 Future Work

The complexity of the problem and the richness of the acquired 3-D and 2-D datasets open a number of future exploration paths along the lines of the system presented in this thesis.

- Range segmentation of non-planar regions and feature extraction of non-linear curves can lead to automated range-to-image registration methods which are applicable to more general scenes. The data reduction due to segmentation can be also achieved with mesh simplification techniques [Guskov *et al.*, 2000]. Mesh simplification can potentially be utilized for increasing the efficiency of solid modeling.

- A very important and hard problem that needs to be addressed in the context of range to image fusion is the blending of brightness images captured from overlapping viewpoints on the 3-D model. Is it possible to perform intelligent blending, especially in the case of blending a sequence of images (video streams) which cover the whole scene? The main question is which sets of images to use when a viewer "sees" the scene from a particular viewpoint. Choosing images which have been captured from locations close to the viewer and also choosing images of higher resolution can lead to high quality render-

ing which can reproduce specular reflections that may exist in the captured scene. A different approach involves the extraction and use of the actual surface albedo maps along the lines of [Bernardini *et al.*, 2000].

- Range-sensing fails to measure transparent surfaces. Also very complicated parts of 3-D scenes (i.e. vegetation) are very hard to model. These areas can be represented only with images (i.e. image-based rendering approaches) and an exciting research opportunity would be in the coexistence of texture-mapped solid models with geometry-less scene representations.

- Another area, which is related to modeling and has not reached its true potential yet, is that of sensor planning. Sensor planning can reduce the amount of sensing operations needed to capture a complete model. In order though to be operational on outdoor scenes, a navigational module in a partial known world is needed. This module is currently under development in our lab [Gueorguiev *et al.*, 2000] and it runs on a mobile robot where the laser range scanner is located (see figure 6.1). The integrated project (called AVENUE) [Allen *et al.*, 2001] contains all the necessary modules (mobile robot navigation, partial 3-D and 2-D maps of the environment of Columbia Campus, sensor planning and site modeling) which can enable a completely autonomous site exploration system. The next section describes the interactive module implemented for the purposes of sensor planning.

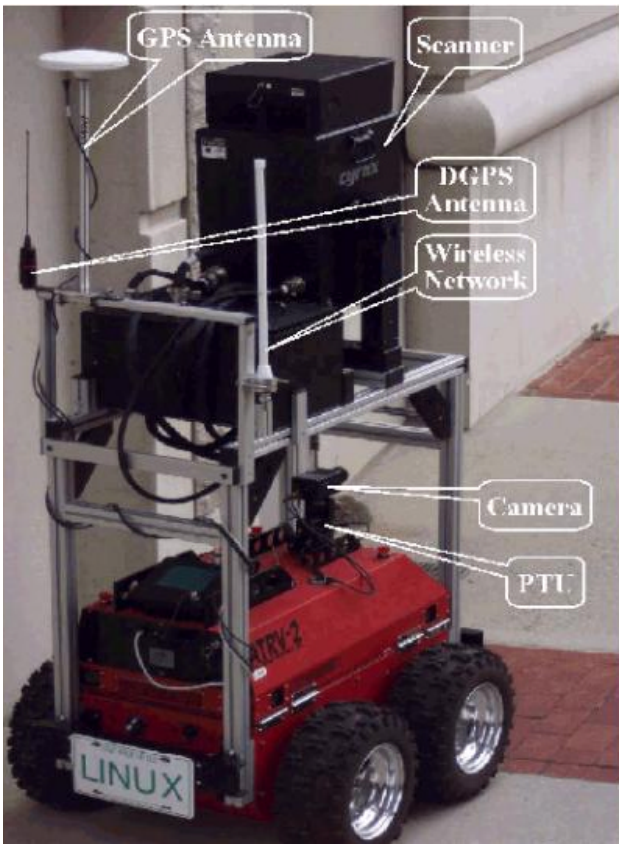

Figure 6.1: Mobile robot used for navigating and exploring urban environments.

6.3 Interactive Sensor Planning

We want to end this thesis with a presentation of our interactive sensor planning module, whose integration with our other algorithms will provide an autonomous range acquisition system for outdoor scenes. In 3-D modeling tasks using range scanning, there is a large cost associated with each sample data acquisition. This includes long acquisition times and very large data sets. Traditionally, most systems that do modeling rely on oversampling the scene, which can be expensive, time-consuming and can still leave gaps where scene elements are missed. The Digital Michelangelo project at Stanford [MICHELANGELO, 2000] is an example of this,

where gigabytes of data scans were acquired, yet small gaps in the range images persist due to the complexity of the scene.

A solution to this problem is to use sensor planning. In cluttered and complex environments such as urban scenes, it can be very difficult to determine where a camera should be placed to view multiple objects and regions of interest. It is important to note that this camera placement problem has two intertwined components. The first is a purely geometric planner that can reason about occlusion and visibility in complex scenes. The second component is an understanding of the optical constraints imposed by the particular sensor (i.e. camera or range sensor) that will affect the view from a particular chosen viewpoint. These include depth-of-field, resolution of the image, and field-of-view, which are controlled by aperture settings, lens size and focal length. To properly plan a correct view, all of these components must be considered.

As part of our goal to automate the model acquisition process, we have prototyped a sensor planning system that can be used to acquire camera views of buildings in cluttered urban environments. We believe this can serve as a first step to creating a true sensor planning system for modeling tasks.

What we have built is an interactive graphical system where sensor planning experiments are performed [Stamos and Allen, 1998]. This system allows us to generate, load and manipulate different types of scenes and interactively select the target features that must be visible by the sensor. The results of the sensor planning experiments are displayed as 3-D volumes of viewpoints that encode the constraints. Virtual sensors placed in those volumes provide a means of synthesizing views in real-time and evaluating viewpoints. In the future, we hope to link this system

up with a mobile robot sensing system that can autonomously navigate to these planned viewpoints [Allen *et al.*, 2001].

6.3.1 Visibility Planning

The computation of the visibility volume involves the computation of the boundary of the *free space* (the part of the 3-D space which is not occupied by objects) and the boundary between the visibility volume and the *occluding volume*, which is the complement of the visibility with respect to the free space. The locus of occlusion–free viewpoints with respect to the 3-D solid model of the scene objects $U = \{u_1, u_2, \ldots, u_m\}$ and the set of target polygonal features $T = \{t_1, t_2, \ldots, t_n\}$ is the visibility volume $V(U,T)$. Each target feature t_i is a 2-D connected part of the scene's boundary $\mathcal{B}(U)$. The complement of the visibility volume with respect to the free space $F(U)$ (open set in space which is not occupied by any objects) is the occluding volume $O(U,T)$, that is $O(U,T) = F(U) - V(U,T)$. Both $O(U,T)$ and $V(U,T)$ are open sets. Details can be found in [Tarabanis *et al.*, 1996, Abrams *et al.*, 1999].

6.3.2 Field of View

A viewpoint which lies in the visibility volume has an unoccluded view of all target features in the sense that all lines of sight do not intersect any object in the environment. This is a *geometric* constraint that has to be satisfied. Visual sensors however impose *optical* constraints having to do with the physics of the lens (Gaussian lens law for thin lens), the finite aperture, the finite extent of the image plane and the finite spatial resolution of the resulting image formed on the image

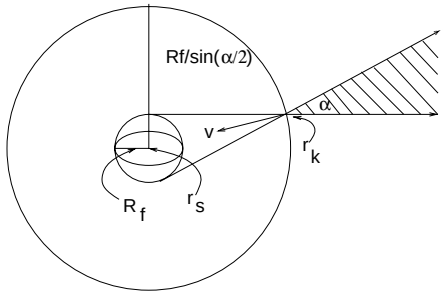

Figure 6.2: Field of view cone (shaded region) for viewing direction $\mathbf{v}$ and field of view angle a. The targets are enclosed in the sphere of radius R_f.

plane, as well as lens distortions and aberrations.

We now consider the *field of view* constraint which is related to the finite size of the active sensor area on the image plane. The targets t_i are imaged if their projection lies entirely on the active sensor area on the image plane. This active sensor area is a 2-D planar region of finite extent. Thus the projection of the target features in their entirety on the image plane depends not only on the viewpoint P_f, but also on the orientation of the camera, the effective focal length and the size and shape of the active sensor area. Those parameters control the position and orientation of the active sensor area in space.

For a specific field of view angle a and a specific viewing direction $\mathbf{v}$ we compute the locus of viewpoints which satisfy the field of view constraint for the set of targets T. If we approximate the set of targets with a sphere S_f of radius R_f and of center $\mathbf{r_s}$ containing them, then this locus is a circular cone $C_{fov}(\mathbf{v}, a, \mathbf{r_s}, R_f)$, called the field of view cone (figure 6.2). The cone axis is parallel to $\mathbf{v}$ and its angle is a. Viewpoints can translate inside this volume (the orientation is fixed) while the targets are imaged on the active sensor area. The locus of the apices of these cones for all viewing directions is a sphere S_{lim} whose center is $\mathbf{r_s}$ and its radius is

$R_f / \sin(a/2)$ (figure 6.2).

6.3.3 Intersecting the Two Constraints

The locus of viewpoints which satisfy more that one constraint is calculated by intersecting the locus of viewpoints which independently satisfy each individual constraint. Integrating the loci calculated in the previous sections we have:

$$I(U, T, \mathbf{v}, a) = V(U, T) \cap C_{fov}(\mathbf{v}, a, T)$$

where $I(U, T, \mathbf{v}, a)$ is the integrated locus (candidate volume), when the viewing direction is $\mathbf{v}$ and the field of view angle is a. Both the visibility volume and field of view cone are represented as solid CAD models. The integrated locus is the result of a boolean operation (intersection) between solids. Intuitively, this solid is the result of the intersection of the visibility volume with a field of view cone. Examples of these regions are given in section 6.3.5.

6.3.4 Interactive System Components

The user interacts with the scene through the graphics models, which can be interactively manipulated. All actions are propagated to the CAD modeler where the boolean operations between models are performed and where the sensor planning algorithms are implemented.

The user selects the target features on the boundary of the scene model and the part of the scene which is going to be the occluding object. First the visibility volume (see section 6.3.1) is computed and displayed overlaid on the scene. After that the user selects a camera orientation $\mathbf{v}$ and a field of view angle a and the

corresponding field of view cone is computed (see section 6.3.2) and displayed. Finally, the intersection of the previous volumes is computed and this is the final result (candidate viewpoints).

The camera selection module allows a virtual camera to be placed and oriented inside the set of *candidate* viewpoints. The camera's field of view angle is interactively set by the user. The resulting image can be used to verify the correctness of the method. Sensor Planning experiments can be performed in real–time using this system.

6.3.5 Experimental Results

We have tested the system using a site model of Rosslyn, Virginia. Our initial input was a textured-mapped VRML model of the city provided by GDE Systems Inc (http://www.gdesystems.com - see Figure 6.3a). Using our model translator we transformed it to a solid CAD model (figure 6.3b) which consisted of 488 solid blocks. We applied the sensor planning algorithms to a part of this model whose boundary consisted of 566 planar faces.

In the first experiment (figure 6.4a) one target (blue face) is placed inside the urban area of interest. The visibility volume is computed and displayed (transparent polyhedral volume). For a viewing direction of $\mathbf{v_1} = (0^o, 22^o, 0^o)$ (Euler angles with respect to the global Cartesian coordinate system) and field of view angle of $a_1 = 44^o$, the field of view locus is the transparent cone on the left. The set of candidate viewpoints $I_1(\mathbf{v_1}, a_1)$ (intersection of visibility with field of view volume) is the partial cone on the left. For a different viewing direction $\mathbf{v_2} = (0^o, 91^o, 0^o)$ the set of candidate viewpoints $I_1(\mathbf{v_2}, a_1)$ is the partial cone on the right.

In the second experiment (figure 6.4b) a second target is added so that two targets (blue planar faces) need to be visible. The visibility volume, the field of view cone for the direction $\mathbf{v_1}$ and the candidate volumes $I_2(\mathbf{v_1}, a_1)$ (left) and $I_2(\mathbf{v_3}, a_1)$ (right) are displayed. The viewing orientation $\mathbf{v_3}$ is equal to $(0^o, 71^o, 0^o)$. The visibility volume and the candidate volume $I_2(\mathbf{v_1}, a_1)$ are subsets of the corresponding ones in the first experiment.

If we place a virtual camera inside the volume $I_1(\mathbf{v_2}, a_1)$, set the field view angle to a_1 and the orientation to $\mathbf{v_2}$, then the synthesized view is displayed on figure 6.5a. The target is clearly visible. Placing a virtual camera outside of the visibility volume (point $(509.92, 41.70, 366.80)$) results in the synthesized view of figure 6.5b. Clearly the target is occluded by one object of the scene. The orientation of the camera is $(0^o, 84^o, 0^o)$ (for every viewpoint outside the visibility volume there does not exist any camera orientation that would result in an unoccluded view of the target). If we place a virtual camera on the boundary of the the candidate volume $I_1(\mathbf{v_2}, a_1)$ (point $(375.59, 52.36, 348.47)$), then in the resulting synthesized view (figure 6.5c) we see that the image of the target is tangent to the image of one object of the scene. Again the camera orientation is $\mathbf{v_2}$ and the field of view angle a_1.

In figure 6.5d we see a synthesized view when the camera is placed on the conical boundary of the candidate volume $I_2(\mathbf{v_3}, a_1)$. The camera's position is $(159.42, 30.24, 347.35)$. The transparent sphere is the sphere S_f (see section 6.3.2) used to enclose the targets. We see that S_f is tangent to the bottom edge of the image, because the viewpoint lies on the boundary of the field of view cone. Finally the figure 6.5e has generated by a camera placed on the polyhedral boundary of

the candidate volume $I_2(\mathbf{v_3}, a_1)$ (position $(254.78, 49.28, 350.45)$).

We have implemented an interactive system where sensor planning experiments can be performed in real-time for complex urban scenes [Stamos and Allen, 1998]. The system can compute visibility and field of view volumes as well as their intersection. The resulting locus consists of viewpoints which are guaranteed to be occlusion–free and places targets within the field of view. Object models and targets can be interactively manipulated and camera positions and parameters selected to generate synthesized images of the targets that encode the viewing constraints. Given site models of scenes, the system can be used to plan view positions for a variety of tasks including surveillance, safety monitoring, and site design.

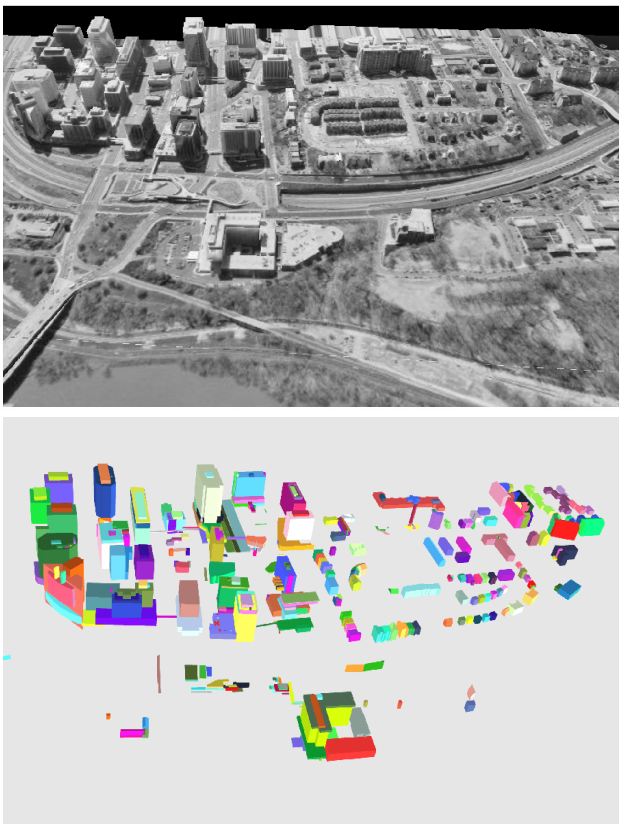

Figure 6.3: a) VRML Graphics model of Rosslyn, VA. b) Solid CAD model computed from the graphics model.

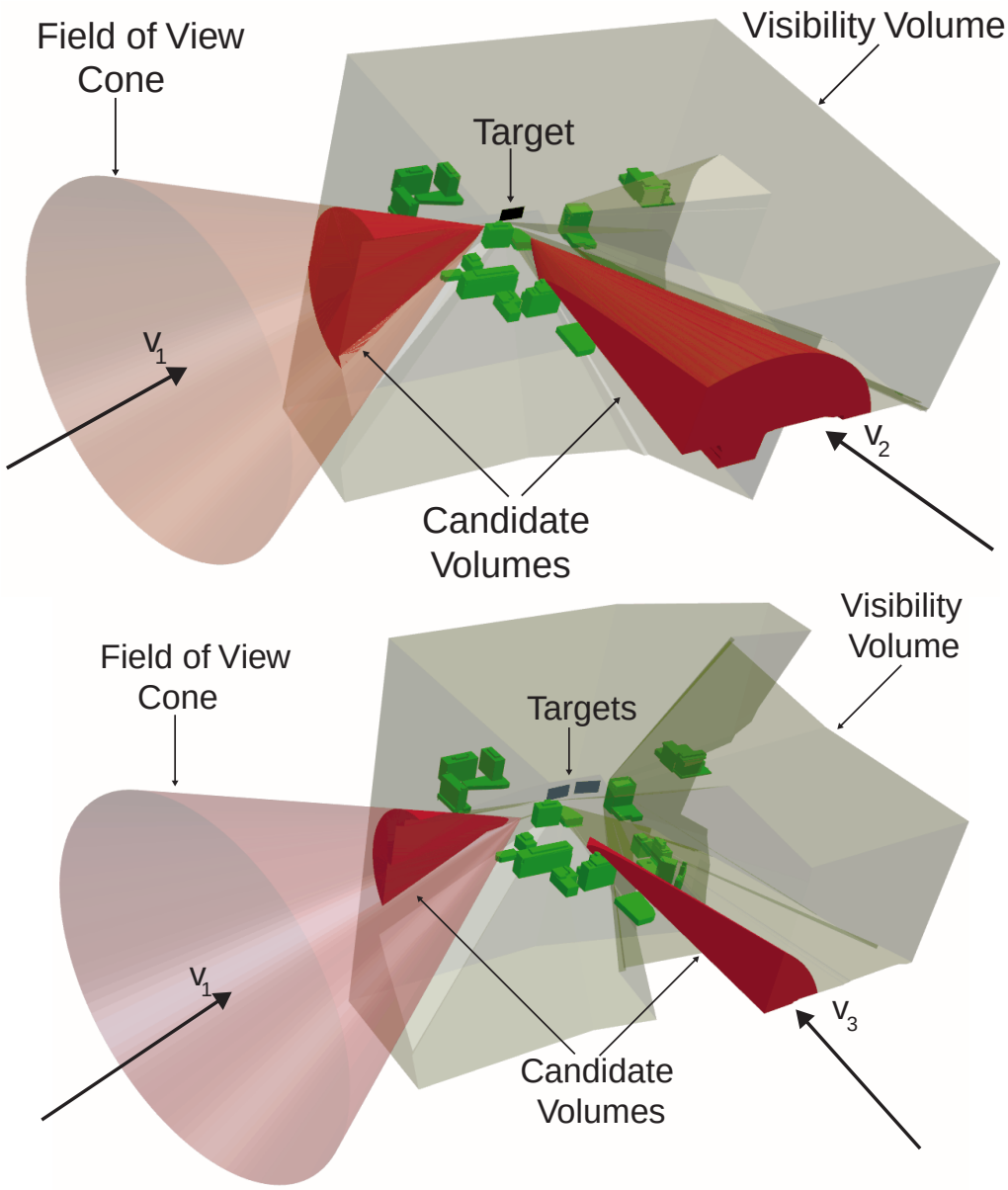

Figure 6.4: Two experiments. a) (top figure) One target and b) (bottom figure) two targets are placed in the urban area. The targets are planar faces. The Visibility Volumes (transparent polyhedral volumes), the Field of View Cones for the direction $\mathbf{v_1}$ (transparent cones) and the Candidate Volumes (intersection of the visibility volumes with the field of view cones) for the viewing direction $\mathbf{v_1}$ (left partial cones) and for the directions $\mathbf{v_2}$ (right partial cone, top figure) and $\mathbf{v_3}$ (right partial cone, bottom figure) are displayed. The Field of View Cones for the directions $\mathbf{v_2}$ (top) and $\mathbf{v_3}$ (bottom) are not shown.

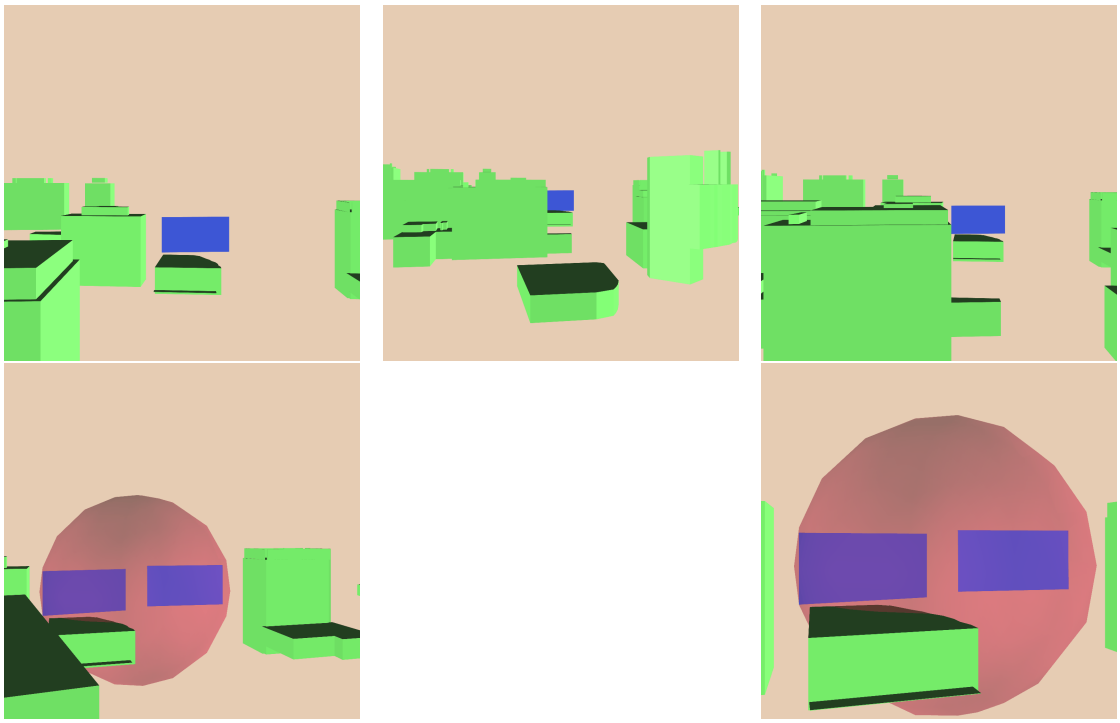

Figure 6.5: Synthesized views. Single target (blue face): the camera is placed a) (clockwise from upper-left) inside the candidate volume, b) out of the visibility volume and c) on the boundary of the candidate volume. Two targets: the camera is placed on d) the conical boundary and e) the polyhedral boundary of the candidate volume.

Bibliography

[Abrams *et al.*, 1999] Steven Abrams, Peter Allen, and Konstantinos Tarabanis. Computing camera viewpoints in an active robot workcell. *International Journal of Robotics Research*, 18(3):267–285, 1999.

[Allen *et al.*, 2001] Peter K. Allen, Ioannis Stamos, Atanas Gueorguiev, Ethan Gold, and Paul Blaer. AVENUE: Automated site modeling in urban environments. In *3rd Int. Conference on Digital Imaging and Modeling*, 2001.

[Alter and Grimson, 1997] T. D. Alter and W. E. L. Grimson. Model-based alignments in the presence of uncertainty. In *CVPR*, pages 344–349, Puerto Rico, 1997.

[Antone and Teller, 2000a] Matthew Antone and Seth Teller. Automatic recovery of camera positions in urban scenes. Technical Report LCS TR-814, MIT, December 2000.

[Antone and Teller, 2000b] Matthew E. Antone and Seth Teller. Automatic recovery of relative camera rotations for urban scenes. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 282–289, Hilton Head, NC, July 2000.

[ARTISAN, 2000] ARTISAN, PROJECT, 2000.
http://www.ri.cmu.edu/projects/project_94.html.

[Baillard and Zisserman, 2000] C. Baillard and A. Zisserman. A plane-sweep strategy for the 3D reconstruction of buildings from multiple images. In *19th ISPRS Conglress and Exhibition*, July 2000.

[Ballard and Brown, 1982] Dana Ballard and Christofer Brown. *Computer Vision.* Prentice-Hall, 1982.

[Becker and Bove, 1995] Shawn Becker and V. Michael Jr. Bove. Semi–automatic 3–D model extraction from uncalibrated 2–D camera views. In *SPIE Visual Data Exploration and Analysis II*, volume 2410, pages 447–461, February 1995.

[Becker, 1997] Shawn Becker. *Vision–assisted modeling from model–based video representations.* PhD thesis, Massachusetts Institute of Technology, February 1997.

[Beraldin *et al.*, 1997] J-A Beraldin, L. Cournoyer, et al. Object model creation from multiple range images: Acquisition, calibration, model building and verification. In *Intern. Conf. on Recent Advances in 3–D Dig. Imaging and Modeling*, pages 326–333, Ottawa, Canada, May 1997.

[Bergevin *et al.*, 1995] R. Bergevin, D. Laurendeau, and D. Poussart. Registering range views of multipart objects. In *Computer Vision and Image Understanding*, volume 61, pages 1–16, 1995.

[Bernardini and Rushmeier, 2000] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. In *Eurographics 2000 State of the Art Report (STAR)*, 2000.

[Bernardini *et al.*, 1999] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. on Vis. and Comp. Graph.*, 5(4):349–359, 1999.

[Bernardini *et al.*, 2000] F. Bernardini, I. Martin, and H. Rushmeier. High-quality texture reconstruction. Technical Report RC 21656, IBM (To appear to IEEE Trans. on Vis. and Comp. Graph.), February 2000.

[Besl and Jain, 1988] Paul J. Besl and Ramesh C. Jain. Segmentation through variable–order surface fitting. *IEEE Trans. Patt. Anal. and Machine Intell.*, 10(2):167–192, March 1988.

[Besl and Mckay, 1992] P. J. Besl and N. D. Mckay. A method for registration of 3–D shapes. *IEEE Trans. Patt. Anal. and Machine Intell.*, 14(2), February 1992.

[Besl, 1988] P. J. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1:127–152, 1988.

[Boyer *et al.*, 1994] Kim L. Boyer, Muhammad J. Mirza, and Gopa Ganguly. The robust sequential estimator: a general approach and its application to surface organization in range data. *IEEE Trans. Patt. Anal. and Machine Intell.*, 1994.

[Canny, 1986] J. Canny. A computational approach to edge detection. *IEEE Trans. Patt. Anal. and Machine Intell.*, 8(6), 1986.

[Caprile and Torre, 1990] B. Caprile and V. Torre. Using vanishing points for camera calibration. *Inter. Conf. on Computer Vision*, 4:127–140, 1990.

[Cass, 1997] T.A. Cass. Polynomial–time geometric matching for object recognition. *IJCV*, 21(1–2):37–61, 1997.

[Chen and Williams, 1993] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH*, pages 279–288, 1993.

[Christy and Horaud, 1999] Stephane Christy and Radu Horaud. Iterative pose computation from line correspondences. *CVIU*, 73(1):137–144, January 1999.

[Chua and Jarvis, 1996] C. Chua and R. Jarvis. 3-D free-form surface registration and object recognition. *Inter. Journal of Computer Vision*, 17:77–99, 1996.

[Coorg and Teller, 1998] Satyan Coorg and Seth Teller. Automatic extraction of textured vertical facades from pose imagery. Technical Report LCS TR-729, Massachusetts Institute of Technology, January 1998.

[Coorg and Teller, 1999] Satyan Coorg and Seth Teller. Extracting textured vertical facades from contolled close-range imagery. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 625–632, Fort Collins, Colorado, 1999.

[Coorg, 1998] Satyan R. Coorg. *Pose Imagery and Automated Three–Dimensional Modeling of Urban Environments*. PhD thesis, Massachusetts Institute of Technology, September 1998.

[Curless and Levoy, 1996] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 303–312, 1996.

[Cyrax Technologies, 2000] Cyrax technologies, 2000. http://www.cyra.com.

[Debevec *et al.*, 1996] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecure from photographs: A hybrid geometry-based and image-based approach. In *SIGGRAPH*, 1996.

[Debevec *et al.*, 1998] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view–dependent image–based rendering with texture–mapping. In *Eurographics Workshop on Rendering*, Vienna, Austria, 1998.

[DeMenthon and Davis, 1995] Daniel F. DeMenthon and Larry S. Davis. Model-based object pose in 25 lines of code. *IJCV*, 15:123–141, June 1995.

[Dhome *et al.*, 1989] Michel Dhome, Marc Richetin, Jean-Thierry Lapresté, and Gérard Rives. Determination of the attitude of 3–D objects from a single perspective view. *IEEE Trans. Patt. Anal. and Machine Intell.*, 11(12):1265–1278, December 1989.

[El-Hakim *et al.*, 1997] S. F. El-Hakim, P. Boulanger, F. Blais, and J.-A. Beraldin. A system for indoor 3–D mapping and virtual environments. In *Videometrics V*, July 1997.

[Faugeras, 1996] Olivier Faugeras. *Three–Dimensional Computer Vision.* The MIT Press, 1996.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381–395, June 1981.

[Fitzgibbon and Zisserman, 1998] A. W. Fitzgibbon and A. Zisserman. Automatic 3D model acquisition and generation of new images from video sequences. In *Proc. of European Signal Processing Conf. (EUSIPCO '98), Rhodes, Greece*, pages 1261–1269, 1998.

[Gandhi and Camps, 1994] T.L. Gandhi and O.I. Camps. Robust feature selection for object recognition using uncertain 2D image data. In *CVPR*, pages 281–287, Seattle, WA, 1994.

[Gueziec and Ayache, 1994] A. Gueziec and N. Ayache. Smoothing and matching 3-D space curves. *Inter. Journal of Computer Vision*, 12(1):79–104, 1994.

[Gueorguiev *et al.*, 2000] Atanas Gueorguiev, Peter K. Allen, Ethan Gold, and Paul Blaer. Design, architecture and control of a mobile site modeling robot. In *Intern. Conf. on Rob. & Aut.*, San Fransisco, April 2000.

[Guskov *et al.*, 2000] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schroder. Normal meshes. In *SIGGRAPH 2000*, August 2000.

[Guy and Medioni, 1997] Gideon Guy and Gerard Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy, 3D data. *IEEE Trans. Patt. Anal. and Machine Intell.*, 19(11):1265–1277, 1997.

[Hausler and Ritter, 1999] G. Hausler and D. Ritter. Feature–based object recognition and localization in 3D-space, using a single video image. *CVIU*, 73(1):64–81, 1999.

[Hoffman and Jain, 1987] R.L. Hoffman and A.K. Jain. Segmentation and classification of range images. *IEEE Trans. Patt. Anal. and Machine Intell.*, 9(5):608–620, 1987.

[Hoover *et al.*, 1996] Adam Hoover, Gillian Jean-Baptise, Xiaoyi Jiang, et al. An experimental comparison of range image segmentation algorithms. In *IEEE Trans. Patt. Anal. and Machine Intell.*, pages 1–17, July 1996.

[Horaud *et al.*, 1989] Radu Horaud, Conio Bernard, and Olivier Leboulleux. An analytic solution to the perspective 4-point problem. *CVGIP*, 47:33–44, 1989.

[Horaud *et al.*, 1997] Radu Horaud, Fadi Dornaika, Bart Lamiroy, and Stephane Christy. Object pose: The link between weak perspective, paraperspective, and full perspective. *IJCV*, 22(2):173–189, 1997.

[Horn, 1987] B. K. P. Horn. Closed–form solution of absolute orientation using unit quartenions. *Journal of the Optical Society of America*, 4(4):629–642, April 1987.

[Horn, 1990] B.K.P. Horn. Relative orientation. *Inter. Journal of Computer Vision*, 4:59–78, 1990.

[Huttenlocher and Ullman, 1990] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *IJCV*, 5(7):195–212, 1990.

[Ikeuchi *et al.*, 1996] K. Ikeuchi, T. Shakunaga, M. Wheeler, and T. Yamazaki. Invariant histograms and deformable template matching for SAR target recognition. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 100–105, 1996.

[Ins, 1999] Institute of Industrial Science(IIS), The Univ. of Tokyo. *Urban Multi–Media/3D Mapping workshop*, Japan, 1999.

[Jacobs, 1997] David W. Jacobs. Matching 3–D models to 2–D images. *IJCV*, 21(1–2):123–153, 1997.

[Jain and Dubes, 1988] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[Jiang and Bunke, 1994] X.Y. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Machine Vision and Applications*, 7(2):115–122, 1994.

[Jiang and Bunke, 1999] Xiaoyi Jiang and Horst Bunke. Edge detection in range images based on scan line approximation. *Computer Vision and Image Understanding*, 73(2):183–199, February 1999.

[Johnson and Hebert, 1997] Andrew Edie Johnson and Martial Hebert. Surface registration by matching oriented points. In *IEEE Conf. Computer Vision and Pattern Recognition*, Puerto-Rico, 1997.

[Johnson and Kang, 1997] Andrew Johnson and Sing Bing Kang. Registration and integration of textured 3–D data. In *International Advances in 3–D Digital Imaging and Modeling*, pages 234–241, Ottawa, Canada, May 1997.

[Jurie, 1999] Frederic Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *CVIU*, 73(3):357–373, March 1999.

[Koch, 1993] R. Koch. Dynamic 3–D scene analysis using synthesis feedback control. *IEEE Trans. Patt. Anal. and Machine Intell.*, 15(6):556–568, June 1993.

[Koch, 1996] Reinhard Koch. Surface segmentation and modeling of 3–D polygonal objects from stereoscopic image pairs. In *Inter. Conf. on Pattern Recognition*, Vienna, Austria, 1996.

[Kumar and Hanson, 1994] Rakesh Kumar and Allen R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *Computer Vision, Graphics and Image Processing*, 60(3):313–342, November 1994.

[LaValle and Hutchinson, 1995] S.M. LaValle and S.A. Hutchinson. A bayesian segmentation methodology for parametric image models. *IEEE Trans. Patt. Anal. and Machine Intell.*, 17(2):211–217, 1995.

[Leonardis *et al.*, 1995] A. Leonardis, A. Gupta, and R. Bajcsy. Segmentation of range images as the search for geometric parametric models. *Inter. Journal of Computer Vision*, 14(3):253–277, 1995.

[Liebowitz and Zisserman, 1998] David Liebowitz and Andrew Zisserman. Metric rectification for persepective images of planes. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 482–488, Santa Barbar, CA, 1998.

[Liu *et al.*, 1990] Yuncai Liu, Thomas S. Huang, and Olivier D. Faugeras. Determination of camera location from 2–D to 3–D line and point correspondences. *IEEE Trans. Patt. Anal. and Machine Intell.*, 12(1):28–37, January 1990.

[Lucchese *et al.*, 1997] L. Lucchese, G. Doretto, and G. M. Cortelazzo. Frequency domain estimation of 3–D rigid motion based on range and intensity data. In *International Conference on Recent Advances in 3–D Digital Imaging and Modeling*, pages 107–112, Ottawa, Canada, May 1997.

[Lumelsky, 1985] Vladimir J. Lumelsky. On fast computation of distance between line segments. *Information Processing Letters*, 21:55–61, 1985.

[Magee *et al.*, 1985] M. J. Magee, B. A. Boyter, C. H. Chien, and J. K. Aggrawal. Experiments in intensity guided range sensing recognition of three–dimensional objects. *IEEE Trans. Patt. Anal. and Machine Intell.*, 7(6), August 1985.

[McAllister *et al.*, 1999] David McAllister, Lars Nyland, Voicu Popescu, Anselmo Lastra, and Chris McCue. Real-time rendering of real world environments. In *Eurographics Workshop on Rendering*, 1999.

[McMillan and Bishop, 1995] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, 1995.

[MICHELANGELO, 2000] Digital Michelangelo Project, 2000. http://graphics.Stanford.EDU/projects/mich/.

[Monga *et al.*, 1991] Olivier Monga, Rachid Deriche, and Jean-Marie Rocchisani. 3D edge detection using recursive filtering: Application to scanner images. *Computer Vision, Graphics and Image Processing*, 53(1):76–87, January 1991.

[Montiel and Zisserman, 2001] J. M. M. Montiel and A. Zisserman. Automated architectural acquisition from a camera undergoing planar motion. In *Int. Symp. on Virtual and Augmented Architecture (VAA01)*, page To appear, Dublin, Ireland, June 2001.

[Neugebauer and Klein, 1999] Peter Neugebauer and Konrad Klein. Texturing 3D models of real world objects from multiple unregistered photographic views. In *Eurographics '99*, volume 18, 1999.

[Newman *et al.*, 1993] T.S. Newman, P.J. Flynn, and A.K. Jain. Model-based classification of quadric surfaces. *Computer Vision, Graphics and Image Processing*, 58(2):235–249, 1993.

[Oberkampf *et al.*, 1996] D. Oberkampf, D. DeMenthon, and L.S. Davis. Iterative pose estimation using coplanar feature points. *CVGIP*, 63(3), May 1996.

[Parvin and Medioni, 1988] B. Parvin and G. Medioni. Adaptive multiscale feature extraction from range data. *Computer Vision, Graphics and Image Processing*, 45:346–356, 1988.

[Pollefeys *et al.*, 1998] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self calibration and metric reconstruction in spite of varying and unkonwn internal camera parameters. In *Inter. Conf. on Computer Vision*, pages 90–96, Bombay, India, 1998.

[Poussart and Laurendeau, 1989] Denis Poussart and Denis Laurendeau. *Advances in Computer Vision*, chapter 3-D Sensing for Industrial Computer Vision, pages 122–159. Springer-Verlag, 1989.

[Pulli *et al.*, 1997] Kari Pulli, Michael Cohen, and Tom Duchamp. View–based rendering: Visualizing real objects from scanned range and color data. In *8th Eurographics Workshop on Rendering*, June 1997.

[Pulli *et al.*, 1998] Kari Pulli, Habib Abi-Rached, Tom Duchamp, Linda G. Shapiro, and Werner Stuetzle. Acquisition and visualization of colored 3–D objects. In *Inter. Conf. on Pattern Recognition*, Australia, 1998.

[Quan and Lan, 1999] Long Quan and Zhondgan Lan. Linear N–point camera pose determination. *PAMI*, 21(7), July 1999.

[Rademacher and Bishop, 1998] Paul Rademacher and Gary Bishop. Multiple-center-of-projection images. In *SIGGRAPH*, 1998.

[Reed and Allen, 1999] M. Reed and P. K. Allen. 3-D modeling from range imagery. *Image and Vision Computing*, 17(1):99–111, February 1999.

[Reed and Allen, 2000] Michael Reed and Peter K. Allen. Constrained-based sensor planning for scene modeling. *IEEE Trans. Patt. Anal. and Machine Intell.*, 22(12):1460–1466, December 2000.

[Reed *et al.*, 1997] Michael Reed, Peter Allen, and Ioannis Stamos. Automated model acquitition from range images with view planning. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 72–77, Puerto Rico, June 1997.

[Sabata *et al.*, 1993] B. Sabata, F. Arman, and J.K. Aggarwal. Segmentation of 3D range images using pyramidal data structures. *Computer Vision, Graphics and Image Processing*, 57(3):373–387, 1993.

[Sequiera *et al.*, 1999] V Sequiera, K Ng, E Wolfart, J. Concalves, and D. Hogg. Automated reconstruction of 3D models from real environments. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54:1–22, 1999.

[Shade *et al.*, 1998] Jonathan Shade, J. Steven Gortler, Li-Wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH*, 1998.

[Shi and Malik, 1997] J. Shi and J. Malik. Normalized cuts and image segmentation. *Computer Vision and Pattern Recognition*, 1997.

[Shum *et al.*, 1998] H.-Y. Shum, Mei Han, and R. Szeliski. Interactive construction of 3D models from panoramic mosaic. In *IEEE Conf. Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998.

[Stamos and Allen, 1998] Ioannis Stamos and Peter Allen. Interactive sensor planning. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 489–494, Santa Barbara, CA, June 1998.

[Stamos and Allen, 2000a] Ioannis Stamos and Peter K. Allen. 3–D model construction using range and image data. In *IEEE Conf. Computer Vision and Pattern Recognition*, volume I, pages 531–536, Hilton Head, SC, July 2000.

[Stamos and Allen, 2000b] Ioannis Stamos and Peter K. Allen. Integration of range and image sensing for photorealistic 3D modeling. In *Inter. Conf. on Robotics and Automation*, pages 1435–1440, San Fransisco, CA, May 2000.

[Stamos and Allen, 2001] Ioannis Stamos and Peter K. Allen. Registration of 3D with 2D imagery in urban environments. In *Inter. Conf. on Computer Vision*, Vancouver, Canada, July 2001.

[Taillandier, 2000] Franck Taillandier. Texture and relief estimation from multiple georeferenced images. Master's thesis, Ecole Polytechnique, September 2000.

[Tarabanis *et al.*, 1996] Konstantinos Tarabanis, Roger Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 18(3):279–292, March 1996.

[Taylor *et al.*, 1989] R.W. Taylor, M. Savini, and A.P. Reeves. Fast segmentation of range imagery into planar regions. *Computer Vision, Graphics and Image Processing*, 45(1):42–60, 1989.

[Teller, 2000] MIT City Scanning Project, 2000.
http://graphics.lcs.mit.edu/city/city.html.

[Trucco and Fisher, 1995] Emanuele Trucco and Robert B. Fisher. Experiments in curvature-based segmentation of range data. *IEEE Trans. Patt. Anal. and Machine Intell.*, 17(2):177–182, 1995.

[Turk and Levoy, 1994] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH*, 1994.

[VIT, 2000] Visual Information Technology Group, Canada, 2000. http://www.vit.iit.nrc.ca/VIT.html.

[Wang and Tsai, 1990] Ling-Ling Wang and Wen-Hsiang Tsai. Computing camera parameters using vanishing–line information from a rectangular parallelepiped. *Machine Vision and Applications*, 3:129–141, 1990.

[Weik, 1997] S. Weik. Registration of 3–D partial surface models using luminance and depth information. In *International Conference on Recent Advances in 3–D Digital Imaging and Modeling*, pages 93–100, Ottawa, Canada, May 1997.

[Wells, 1997] W.M. Wells. Statistical approaches to feature–based object recognition. *IJCV*, 21(1–2):63–98, 1997.

[Yu, 2000] Yizhou Yu. *Modeling and Editing Real Scenes with Image-Based Techniques*. PhD thesis, UC Berkeley, 2000.

[Zhang and Faugeras, 1994] Zhengyou Zhang and Olivier Faugeras. Finding planes and clusters of objects from 3D line segments with application to 3D motion determination. *Computer Vision, Graphics and Image Processing*, 60(3):267–284, 1994.

[Zhang, 1993] Y. J. Zhang. Quantitative study of 3D gradient operators. *Image and Vision Computing*, 11(10):611–622, December 1993.

[Zhao and Shibasaki, 1999] Huijing Zhao and Ryosuke Shibasaki. A system for reconstructing urban 3D objects using ground–based range and CCD sensors. In

Urban Multi–Media/3D Mapping workshop, Inst. of Industr. Sc., The Univ. of Tokyo, 1999.

Appendix A

Camera Model and Line Representation

This section describes the projection model that we assume for our camera sensor. The 2–D camera follows the perspective projection model as shown in figure A.1. The effective focal length of the camera is f (expressed in pixel units) and the principal point is $\mathbf{Pp} = (p_x, p_y)$. 3–D scene points $\mathbf{P_i}$ are projected on 2–D image points $\mathbf{p_i}$, and 3–D scene linear segments $\mathbf{L} = (\mathbf{P_i}, \mathbf{P_j})$ are projected on 2–D image linear segments $\mathbf{l} = (\mathbf{p_i}, \mathbf{p_j})$. The 3–D to 2–D projection of the line segment $\mathbf{L}$ to the line segment $\mathbf{l}$ can be mathematically described as: $\mathbf{l} = \mathcal{P}(\mathbf{L})$, where

$$\mathcal{P} = \mathcal{P}(R, \mathbf{T} | f, \mathbf{Pp}).$$

That means that the projection mapping $\mathcal{P}$ depends on the relative position of the range and image sensors $(R, \mathbf{T})$ and on the internal calibration camera parameters $(f, \mathbf{Pp})$. We assume that our image has been corrected with respect to radial distortion effects. We also assume that our range sensor provides accurate 3–D

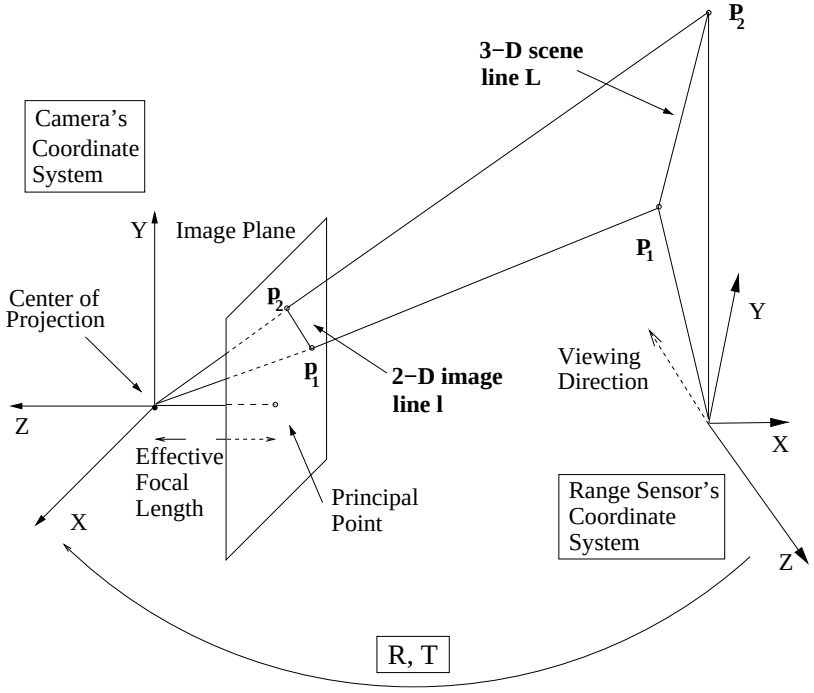

Figure A.1: Camera model

positions of sampled 3–D points with respect to its coordinate system (figure A.1).

We represent 2–D points and 2–D lines as antipodal-points on the Gaussian sphere (figure A.2b). In this manner we can represent 2–D points at infinity[1]. Let $\mathbf{C\tilde{O}P}$ be the center of projection of our camera (figure A.2a). A 2–D point $\mathbf{p_i}$ can be represented by the 3–D unit vectors $\pm\mathbf{n_i}$ (note that there is a sign ambiguity). That means that 2–D points map to pairs $\{(\phi,\theta),(\phi+\pi,-\theta)\}$ of antipodal points on the Gaussian sphere (figure A.2b). We can use a similar representation for 2–D lines: the line $\mathbf{l_{12}}$ connecting the points $\mathbf{p_1}$ and $\mathbf{p_2}$ can be uniquely represented by the 3–D unit vectors $\pm\mathbf{N_{12}} = \pm(\mathbf{n_1} \times \mathbf{n_2})$ which correspond to the normals of the plane $< \mathbf{p_1}, \mathbf{p_2}, \mathbf{C\tilde{O}P} >$. There is a one-to-one correspondence between 2–D points and antipodal-points on the Gaussian sphere. The same holds for 2–D lines.

[1]You can view those points as the of intersection of parallel 2–D lines on the image plane.

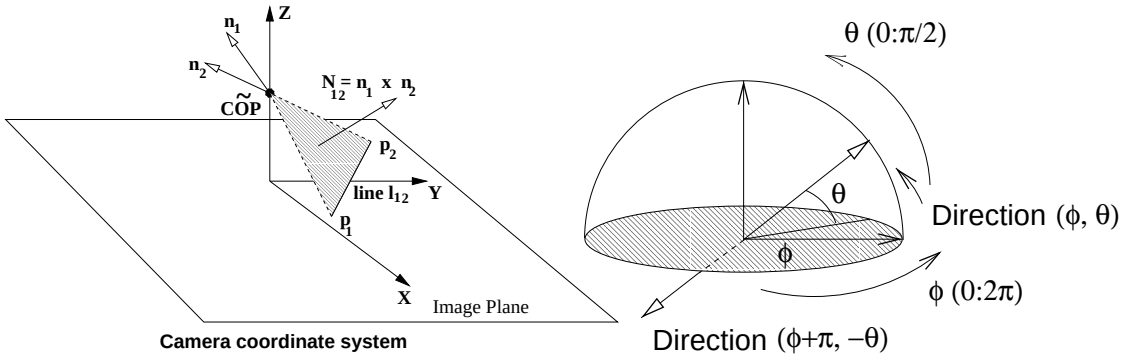

Figure A.2: a) Representing 2–D points and lines as antipodal-points on the Gaussian sphere. A point is represented by the pair of unit vectors $\pm\mathbf{n_i}$ and a line by the pair $\pm\mathbf{n_1} \times \mathbf{n_2}$. b) A point in the Gaussian sphere can be represented by the direction (ϕ, θ) (latitude, longitude). A pair of antipodal points correspond to the directions $d_1 = (\phi, \theta)$ and $d_2 = (\phi + \pi, -\theta)$, where $0 \leq \phi \leq 2\pi$ and $0 \leq \theta \leq \pi/2$. The upper half of the Gaussian sphere is enough for the representation of all pairs of antipodal points.

Thus a 2–D point and a 2–D infinite line can be represented by pairs of the form (ϕ, θ), where $0 \leq \phi \leq 2\pi$ and $0 \leq \theta \leq \pi/2$. This representation is crucial in the algorithms that are described in chapter 5. Note, that the point $\mathbf{C\tilde{O}P}$ does not need to be the exact center of projection. The mapping from points and lines to directions on the Gaussian sphere holds for all $\mathbf{C\tilde{O}P} \neq \mathbf{0}$.

Appendix B

Pose Estimation from Line Matches

Pose estimation from a set of potentially matched 3-D and 2-D structures can be translated to matching the linear borders of those structures. We adapted the algorithm proposed by Kumar and Hanson [Kumar and Hanson, 1994] for the registration between range and 2–D images when a set of corresponding 3–D and 2–D lines is given. The internal calibration parameters of the camera are assumed known since they have been computed by the extracted vanishing points (section 5.3.2). A full optimization with respect to both rotation and translation is performed for the computation of the final pose (section 5.6). During the search for a coarse pose estimate (section 5.5) the pose is optimized with respect to translation only (since the rotation has been already computed). That involves the solution of a linear system of equations.

Let $\mathbf{N_i}$ be the normal of the plane formed by the ith image line and the center of projection of the camera (figure B.1). This vector is expressed in the

coordinate system of the camera. The sum of the squared perpendicular distance of the endpoints $\mathbf{e_i^1}$ and $\mathbf{e_i^2}$ of the corresponding ith 3–D line from that plane is

$$d_i = (\mathbf{N_i} \cdot (R(\mathbf{e_i^1}) + \mathbf{T}))^2 + (\mathbf{N_i} \cdot (R(\mathbf{e_i^2}) + \mathbf{T}))^2, \tag{B.1}$$

where the endpoints $\mathbf{e_i^1}$ and $\mathbf{e_i^2}$ are expressed in the coordinate system of the range sensor. The error function we wish to minimize is

$$E_1(R, \mathbf{T}) = \Sigma_{i=1}^{N} d_i. \tag{B.2}$$

This function is minimized with respect to the rotation matrix R and the translation vector $\mathbf{T}$. This error function expresses the perpendicular distance of the endpoints of a 3–D line from the plane formed by the perspective projection of the corresponding 2–D line into 3–D space (figure B.1). The exact location of the endpoints of the 2–D image segment do not contribute to the error metric and they can move freely along the image line without affecting the error metric. In this case we have a matching between infinite image lines and finite 3–D segments.

The computation of the translation, when the rotation is already known, involves the solution of a linear system of three equations in three unknowns. The full optimization of the metric is explained in the following section.

B.1 Full Optimization

The minimization of that metric is similar to the iterative technique proposed by Horn [Horn, 1990]. Let $\mathbf{e_i}' = R\mathbf{e_i}$, where $\mathbf{e_i}$ is a 3–D point expressed in the coordinate system of the range sensor. Then an incremental infinitesimal rotation $\mathbf{d}\omega$ will transform $\mathbf{e_i}'$ to

$$\mathbf{e_i}'' = \mathbf{e_i}' + \mathbf{d}\omega \times \mathbf{e_i}'. \tag{B.3}$$

Using this fact the application of an infinitesimal incremental rotation $\mathbf{d}\omega$ and an incremental translation $\mathbf{dT}$ would change the error metric to

$$E_1(R\,R(\mathbf{d}\omega), \mathbf{T} + \mathbf{dT}). \tag{B.4}$$

By taking the derivatives of this error with respect to $\mathbf{d}\omega$ and $\mathbf{dT}$ and setting the results equal to 0 we reach a linear system of 6 equations with 6 unknowns (the elements of $\mathbf{d}\omega$ and $\mathbf{dT}$). The solution of this system ($\mathbf{d}\omega$, $\mathbf{dT}$) provides updates for the rotation matrix R and the translation vector $\mathbf{T}$. The rotation is represented as a unit quartenion in order to convert non-infinitesimal rotational estimates $\mathbf{d}\omega$ to valid rotational representations. That procedure is run iteratively until the error metric becomes smaller than a threshold or a maximum number of iterations is reached.

The results are very accurate when there are no mismatches between 3–D and 2–D lines. The extraction of reliable and accurate 3–D and 2–D features is very important for the accuracy of the final registration.

The system of equations has a solution if at least 3 corresponding 3–D and 2–D lines are given. Those lines should not be coplanar and should not meet at the same point in space. In the latter case there are infinite number of solutions for the translation.

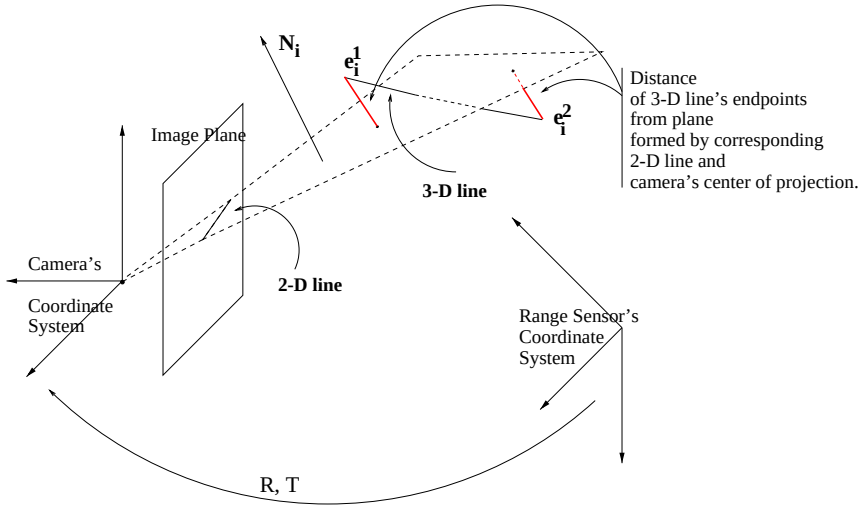

Figure B.1: Error metric in pose estimation