Automated 3D Model Generation for Urban Environments

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät für Elektrotechnik und Informationstechnik der

Universität Fridericiana Karlsruhe

genehmigte

DISSERTATION

von

Dipl.-Ing. Christian Früh

aus

Herbolzheim

Tag der mündlichen Prüfung: 29. Oktober 2002

Hauptreferent:	Prof. Dr.-Ing. K.D. Müller-Glaser Universität Karlsruhe
1. Korreferent:	Prof. Dr. A. Zakhor University of California at Berkeley
2. Korreferent:	Prof. Dr. W. Heering Universität Karlsruhe

Karlsruhe, den 20. September 2002

Abstract

In this thesis, we present a fast approach to automated generation of textured 3D city models with both high details at ground level and complete coverage for bird's-eye view. A ground-based facade model is acquired by driving a vehicle equipped with two 2D laser scanners and a digital camera under normal traffic conditions on public roads. One scanner is mounted horizontally and is used to determine the approximate component of relative motion along the movement of the acquisition vehicle via scan matching; the obtained relative motion estimates are concatenated to form an initial path. Assuming that features such as buildings are visible from both ground-based and airborne view, this initial path is globally corrected by Monte-Carlo Localization techniques using an aerial photograph or a Digital Surface Model as a global map. The second scanner is mounted vertically and is used to capture the 3D shape of the building facades. Applying a series of automated processing steps, a texture-mapped 3D facade model is reconstructed from the vertical laser scans and the camera images. In order to obtain an airborne model containing the roof and terrain shape complementary to the facade model, a Digital Surface Model is created from airborne laser scans, then triangulated, and finally texture-mapped with aerial imagery. Finally, the facade model and the airborne model are fused to one single model usable for both walk- and fly-thrus. The developed algorithms are evaluated on a large data set acquired in downtown Berkeley, and the results are shown and discussed.

Erklärung

Ich versichere wahrheitsgemäß, die Dissertation bis auf die angegebenen Hilfen selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer und eigenen Veröffentlichungen unverändert oder mit Änderungen entnommen wurde.

Berkeley, USA, den 19. September 2002

Acknowledgements

This Ph.D. thesis has been developed during my stay at the Video and Image Processing Lab of the University of California at Berkeley. Therefore, I am deeply grateful to my adviser Prof. A. Zakhor, who provided me with the unique opportunity to do research at one of the most prestigious universities of the United States. Her excellent advice and active support has provided the foundation for this work, but most importantly, she always gave me the freedom and encouragement to explore new ideas.

Furthermore, my adviser Prof. K.-D. Müller-Glaser of the University of Karlsruhe deserves my special gratitude for his willingness to serve as my adviser, and his interest in this work. Despite his tight schedule, he allocated the time to review my work without delay, and provided me with many useful comments and suggestions through personal meetings and via phone and email. Likewise, I would like to thank Prof. W. Heering for following my work from the very beginning and serving on my thesis defense committee.

Furthermore, I would like to thank Prof. R. Dillmann and my former colleagues from the Institute for Process Control, Robotics and Automation, where I gained useful experience in robotics and laser scanning. I also owe my very special thanks to my fellow students at U.C. Berkeley's Video and Image Processing Lab, in particular to John Flynn, Samson Cheung, Thinh Nguyen, and Vito Dai; surely my work would have proceeded at slower pace without their help and friendly support. Finally, I would like to express my gratitude to the American taxpayers, who have financed this work via the Army Research Office of the U.S. Department of Defense.

But lastly, none of this would have been possible without my parents. Their love, care and support have shaped my character and life, and their moral and financial support has enabled my education.

Danksagung

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Video and Image Processing Lab der University of California in Berkeley, USA. Daher gilt mein besonderer Dank zunächst meiner Korreferentin Prof. Dr. A. Zakhor, die mir das Forschen an einer der renommiertesten Universitäten der USA ermöglicht hat. Ihre freundschaftlichen Ratschläge, Anregungen und Korrekturen haben diese Arbeit maβgeblich geprägt.

Des Weiteren gilt mein besonderer Dank meinem Hauptreferenten Prof. Dr.-Ing. K.-D. Müller-Glaser, der aus Interesse an dieser Arbeit die Betreuung seitens der Universität Karlsruhe übernommen hat und sich trotz vieler Termine die Zeit nahm, mir mit Rat und Tat zur Seite zu stehen und die Begutachtung ohne Verzögerung durchzuführen. Es gibt sicher nur wenige Professoren, die sich in derart freundschaftlicher und herzlicher Art für ihre Doktoranden einsetzen. Dies gilt ebenso für meinen zweiten Korreferenten Prof. Dr. W. Heering, der die Arbeit von Anfang an verfolgt und mich stets gut beraten hat.

Ferner bedanke ich mich bei allen anderen, die zum Gelingen dieser Arbeit beigetragen haben. Als da wären Prof. Dr. R. Dillmann und meine Kollegen vom Institut für Prozessrechentechnik, Automation und Robotik, wo ich wertvolle Erfahrungen im Bereich Robotik und Laserscannen sammeln konnte, die mir bei dieser Arbeit von groβem Nutzen waren. Ferner natürlich bei meinen Kollegen und Studenten in Berkeley, insbesondere John Flynn, Samson Cheung, Thinh Nguyen, and Vito Dai, ohne deren freundschaftliche Ratschläge, Unterstützung und Know-how meine Arbeit sicherlich viel langsamer vorangegangen wäre. Mein Dank gilt schlieβlich auch dem amerikanischen Steuerzahler, der in Gestalt des Army Research Office des U.S. Department of Defense diese Arbeit finanziell unterstützt hat.

Alles wäre jedoch nicht möglich gewesen ohne meine Eltern, deren Liebe und Erziehung mich geprägt haben und deren moralische und finanzielle Unterstützung mein Studium ermöglicht hat.

Table of Contents

Table of Symbols

θ	yaw angle
n	scan index
S_n	scan
υ	scan direction angle
$s_{n,\upsilon}$	scan point
Q	quality of match
$\Delta\psi$	angular spacing of scan measurements
$\vec{V}_T$	translatory speed
Ω_T	angular speed
c	congruence coefficient
π	pose state
bel	belief probability density
a_t	action data
o_t	observation data
w_i	normalized importance factor
d_{crit}	critical distance
γ	split depth

Table of Abbreviations

2D	Two-dimensional
3D	Three-dimensional
API	Application Programming Interface
CAD	Computer-Aided Design
CCD	Charge Coupled Device
DSM	Digital Surface Model
DTM	Digital Terrain Model
GPS	Global Positioning System
ICP	Iterative Closest Point
INS	Inertial Navigation System
LASER	Light Amplification by Stimulated Emission of Radiation
LIDAR	Light Detection and Ranging
LOD	Level of Detail
MCL	Monte-Carlo Localization
mrad	milliradians
RAID	Redundant Array of Inexpensive Disks
RANSAC	Random Sampling and Consensus
VRML	Virtual Reality Modeling Language

Table of Figures

1 Introduction

The problem of capturing and displaying real-world objects has been addressed in many ways. From the first paintings in stone-age time to modern digital cameras, two-dimensional pictures have been the most important means to grasp and share a piece of experienced reality, and these forms of visualization have been greatly extended with the invention of movies as a series of images. However, pictures and movies cannot truly reproduce the impression of a real world experience, because they are passive, i.e. one can only re-view objects from a predefined pose or trajectory. This is in contrast to the human desire to examine objects by turning them around or to explore environments by moving in them without restrictions. In computer science, the field of virtual reality has emerged to satisfy this need, providing a graphical 3D interface between computer and user. Since the graphics capability of computer systems has increased by orders of magnitude during the last decade, the availability of affordable rendering power has opened the door for a variety of new graphics applications, and has at the same time created an enormous interest in three-dimensional models of objects or environments.

In this context, the necessity for capturing three-dimensional (3D) models of urban environments has been growing steadily during recent years. While 3D models, in combination with a rendering system, enable the reconstruction of arbitrary views, their usability extends far beyond the mere interactive replication of an environment; rather, enhanced with all kinds of additional information, they are useful in a variety of applications. In architecture and urban planning, the appearance of buildings in an existing urban setting can be simulated before the actual construction, in order to verify compatibility not solely based on human imagination. In computer gaming, the immersive experience can be substantially increased if a game takes place in environments familiar to the user, e.g. in his hometown, rather than in an artificial setting. In the entertainment industry, movies are increasingly based on virtual 3D models, providing special effects that are difficult if not impossible to obtain with other technologies. The usage of 3D city models has for example enabled “Mission Impossible II” (1996), “The Matrix” (1999), and “The Matrix Reloaded” (to appear 2003) to set new milestones in the movie industry. And as an unfortunately recently emerging application, 3D models are used for simulations of urban disaster or terrorism scenarios. According to Planet9, a San Francisco-based virtual reality company specializing in selling hand-made 3D city models, the Washington, DC and New York City models have been their most heavily used ones in the aftermath of the September 11th terrorist attacks.

This list can be continued with many other potential applications such as 3D displays for car navigation units, 3D city maps, and simulation of radio wave propagation for the cell phone industry, all of which need accurate 3D city models. The requirements with regard to the geometric level of detail and the photo-realistic appearance differ from application to application. For a far-view fly-thru, it is sufficient to have a coarse approximation of the building tops and the terrain, whereas for a close-up walk- or drive-thru, it is essential to have a high-resolution model of the building facades. And while even the geometry alone may already be sufficient for purposes such as the simulation of radio wave

propagation, the photo-realism requirements demanded by visualization applications are generally enormously high: persons familiar with an environment are extremely sensitive to changes and do immediately notice any discrepancy between the virtual model and known reality. It is by no means adequate for this purpose to utilize an artificial model, simply composed of repetitive geometry and texture.

Hence, the problem of acquiring a sufficiently detailed and photo-realistic model of a city has to be solved. Existing methods for model acquisition are complicated and time consuming, and the lack of photo-realistic models at affordable costs and acceptable acquisition time is prohibitive to the broad usage of virtual worlds. In the past, there have been various attempts to create large scale city models in an automated or semi-automated way from airborne view, either using stereo vision approaches on aerial or satellite images, or, more recently airborne laser scans. Both approaches have the disadvantage that their resolution is low, and more importantly, only the roofs of the buildings are captured, but not the facades. This essential disadvantage prohibits their use in photo-realistic walk- or drive-thru applications, which require an enormous level of detail at ground level.

Previous approaches to acquiring highly detailed models from ground-based view, however, are commonly still manual, e.g. based on entering coordinates from construction plans, or at best semi-automated, e.g. using photos and geometric primitives, which are combined to a model by manually selecting correspondences. None of the existing approaches scales for an entire city, since it would typically take months to create such a large model, due to the significant manual intervention. This process is not only prohibitively expensive, but is also unsuitable in applications where the goal is to monitor changes over time, for example detecting damage or possible danger zones after natural disasters such as earthquakes, land slides or hurricanes, or documenting construction progress for a building. In order to enable the broad use of urban virtual reality, photo-realistic model acquisition has to become a task as simple as taking a picture or recording a video, which can be done without sophisticated expert knowledge in acceptable time.

In this dissertation, we will address the model acquisition problem by proposing an innovative approach to capturing ground-based facade models of urban environments. In this approach, the 3D geometry of a city is acquired using a combination of inexpensive 2D laser scanners, mounted on a pickup truck at a 90-degree angle towards each other, and texture is acquired using a synchronized digital camera. This acquisition vehicle moves at normal speeds on public roads, and since data is acquired continuously rather than in a stop-and-go fashion, our method is extremely fast. The collected data is processed offline, and 3D models of the building facades are generated completely automatically. Additionally, we show that it is possible to merge these facade models with data from airborne view, in order to create a complete 3D model, containing facades as well as building tops and terrain shape.

In this thesis, we show that our approach complies with the following key objectives:

- **Automatism** – no human intervention is necessary.
- **Photo-realism** – rendering the created building models yields images that are visually pleasing and approach the quality of a photo.
- **Speed** – both data acquisition and automated model generation are extremely fast.
- **Scalability** – the complexity is linear and computation time increases only proportional to the covered area.

1.1 Contributions of this Dissertation

For the various aspects to be solved in the context of automated 3D modeling, we employ methods from different research areas such as mobile robotics, optical measurement techniques, computer vision, and computer graphics. The interdisciplinary nature of model acquisition and rendering requires the broad scope that this dissertation comprises.

Specifically, the main contributions of this dissertation are:

- **Innovative data acquisition approach**. We suggest a new approach capable of acquiring detailed facade models at unprecedented speed, since data is acquired continuously, rather than in a stop-and-go fashion.
- **Development of a mobile acquisition system**. Mounted on top of a pickup truck, our acquisition system is equipped with 2D laser scanners and a digital camera. All devices are synchronized, and we have developed real time software to handle and time stamp the incoming data streams.
- **Adaptation of indoor robot localization algorithms to outdoor environments.** In order to provide the level of localization precision necessary for the proposed acquisition approach, we have adapted methods previously only utilized for indoor mobile robots to our high-speed vehicle and the city environment. In particular, the adapted methods are map generation, scan matching, and Monte Carlo Localization.
- **3D processing algorithms.** The different nature of our data requires new 3D processing algorithms, since traditional methods of 3D scan processing are not applicable. We have developed a framework of scalable algorithms for the entire processing pipeline, addressing foreground removal, facade geometry reconstruction, texture mapping, and optimization for rendering
- **Registration and merging of ground-based and airborne models.** We have developed methods to register and augment the facade models obtained from our innovative ground-based data acquisition with data from airborne view, in order to obtain building tops, terrain shape, and hence a complete model of an urban environment.

The developed methods and their results have been covered in publications and presentations in international conferences: In [Früh et al., 2001], we have introduced our

acquisition approach for the first time. The tracking aspect of our precise localization approach is described in [Früh and Zakhor, 2001 a], whereas [Früh and Zakhor, 2001 b] focuses on the global correction and registration in respect to a global map. Finally, [Früh and Zakhor, 2002] details the 3D data processing and model generation.

1.2 Organization of this Dissertation

This thesis is organized as follows:
Chapter 2 gives an overview of background and related work in three-dimensional model generation for architectural structures. Both automated and interactive methods are summarized, and advantages and shortcomings are discussed. In **Chapter 3**, we introduce our new model acquisition approach, and describe hardware and software that has been developed for the data acquisition process.

A variety of new aspects and problems occur in conjunction with our new method. In particular, it turns out that one of the most essential problems is the accurate localization of the vehicle and its sensors, in other words the reconstruction of the pose during data acquisition and the assignment to individual laser scans and camera images. Chapters 4 and 5 address this localization problem: **Chapter 4** focuses on tracking the vehicle based on relative pose estimates. These relative poses are obtained from scan-to-scan matching of subsequent horizontal laser scans and concatenated to an initial acquisition path estimate. In **Chapter 5** we propose innovative methods to globally correct pose using Monte-Carlo-Localization, a probabilistic localization approach recently developed in mobile robotics. The initial path is corrected in respect to a global edge map, which we derive either from aerial photos or from a digital surface model obtained from airborne laser scans.

Chapter 6 is devoted to the completely automated offline processing of the vertical scans and the camera images, and the reconstruction of a 3D facade model. We describe methods of handling the large-size data efficiently, of identifying and removing foreground objects such as trees, and of reconstructing geometric triangular facade models. Furthermore, we devise algorithms to texture-map the architectural structures and to optimize these models for interactive rendering. In **Chapter 7**, we describe the usage of airborne laser scans and aerial images to create an airborne surface mesh containing the parts missing in the facade models, such as rooftops and terrain shape. We merge the complementary models to obtain a final complete model, suitable for both walk- and fly-thrus.

In **Chapter 8**, we evaluate all proposed methods for a large data set of downtown Berkeley. The results are shown and discussed, and the computational performance and complexity is analyzed. **Chapter 9** finally concludes this thesis with a summary and suggestions for future work.

2 Background and Related Work

In this chapter, we describe the background and context of this thesis and introduce terms and concepts commonly used in the context of 3D model generation and rendering. We also summarize technologies and existing approaches for 3D model acquisition and analyze their advantages as well as their shortcomings.

2.1 3D Model Representation and Rendering

The problem of capturing and displaying real-world objects has captured human's interest since ancient time, from paintings and sculptures to photographs and movies. Today, the advances in computer technology have started to overcome the restriction of only passively viewing pre-recorded 2D images and have made it possible to provide more realistic impressions by immersing into 3D dimensional worlds to be explored interactively. In computer science, the field of virtual reality has emerged to satisfy this need, providing a graphical 3D interface between computer and user. In order to enable the immersion of a person in an environment, an interactive display is necessary, which allows rendering, i.e. painting, of an object or environment from an arbitrary viewpoint upon demand.

An object or environment can be defined by a geometrical model, which is, according to the definition in the Oxford English Dictionary, "a representation in three dimensions of some projected or existing structure, or of some material object artificial or natural, showing the proportions and arrangement of its component parts". While this representation can be volumetric, such as polyhedral or voxel-based for applications such as medical imaging or FEM simulations, it is usually surface-based for rendering solid objects. In other words, for the purpose of visualization and rendering, a 3D model contains the geometry and the visual appearance of its visible surfaces.

For the vast majority of models, surface geometry representation is based on triangles. Defined by three vertices only, a triangle is the most basic 2D primitive, and all other polygonal primitives, e.g. quadrilaterals, can be composed from them. To represent a surface in 3D space, triangles are concatenated to a mesh, in which adjacent triangles share vertices. While vertex coordinates and connectivity define the geometry, additional attributes specify the surface properties and thus, in conjunction with a lighting model of the environment, the visual appearance during rendering. The most important attributes are surface normals, vertex normals, and material properties such as shininess, reflectance, and color or texture. In the context of Computer Vision, texture means often a small sample of a larger stochastic pattern, sufficient to describe the stochastic properties. In the early days of Computer Graphics, when graphics memory was exiguous, small texture pieces were used in a repetitive manner; however, since available graphics memory and hence possible sample size have increased by orders of magnitude, entire non-repetitive images can be used as texture instead of small pattern samples. Therefore, in Computer Graphics, texture mapping is now commonly understood as attaching an arbitrary 2D image to geometrical primitives, without making restrictions

about its content; essentially, it is "wrapping" an image around a geometric structure. In order to make 3D models look photo-realistic, i.e. to make them (ideally) indistinguishable from a camera snapshot, it has consequently become popular to use real-world imagery as texture. Essentially, the image formation process in the camera is inverted, and the picture's color and texture information is back-projected onto the geometry.

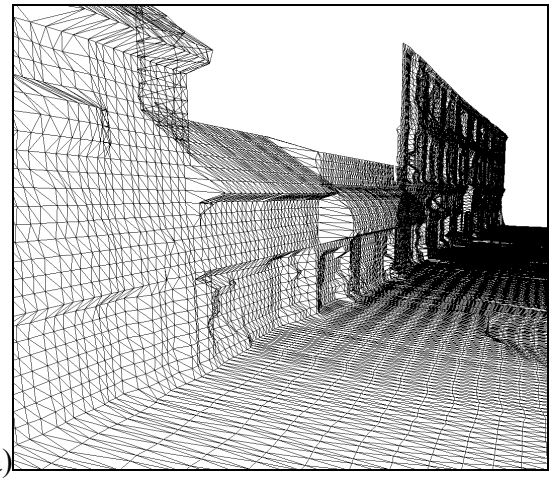
a)  b) 

Figure 2-1: 3D building facade model based on a triangular mesh; (a) mesh displayed as wireframe, (b) same mesh texture mapped with camera images

In recent work, there have also been interesting attempts to introduce point-based instead of triangular model representations. An example is Q-splatting [Rusinkiewicz and Levoy, 2000], in which the geometry and texture is represented by a set of 3D points, with associated color and normal vector for each point. This approach appears promising for cluttered structures that are difficult to approximate by smooth surfaces, and hence result in inadequately large triangular meshes. However, for most objects and in particular for buildings, smooth or planar surfaces are common, thus making triangle-based representations far more efficient than point-based representations.

Using an explicit 3D model is not the only possibility of providing a 3D real-world impression to a human: in fact, the virtual replication of an environment for a walk-thru can be reduced to simply showing the correct image for a given position. Accordingly, a second popular visualization approach, which generally relies more on acquired images than on 3D geometry, has been developed. In this approach, called image-based rendering, it is assumed that correct texture distracts the eye from geometric imperfections if a viewpoint is sufficiently close to the one from which an image was taken. To render a view, appropriate images from nearby locations are retrieved from a large database and combined via warping and mosaicing. While this step typically utilizes 3D information, it is sufficient to possess only very approximate knowledge about the 3D geometry of a scene. In a full virtual reality setup, it is possible to create a true 3D impression for a human user even without any knowledge about its 3D content by showing each eye a slightly different view of the scene. Although humans perceive a

3D world, they observe in fact only a pair of two-dimensional images. It is the brain that combines the two images to a 3D impression by exploiting small disparities due to the different viewpoints in a process called stereo vision.

Since they are capable of including each detail of a scene without a merely impossible 3D reconstruction, image-based rendering techniques are able to achieve an impressive level of photo-realism. The visual quality of image-based rendering outperforms model-based rendering, if both (a) acquired images close to the rendered view are available, and (b) the object geometry is too complex to be properly represented by a geometric model, e.g. in case of trees or panoramic views over a large urban area. The main reason is that for these situations, model-based techniques explode in their necessary number of polygons, while the average display size of a polygon on the screen can fall below few pixels. In contrast, image based rendering has inherently an appropriate ratio between source image size and rendered size, since it performs roughly a one-to-one copy to the screen.

However, the various types of image-based techniques have similar severe drawbacks that prohibit their use for a wide range of applications. First, the data size is orders of magnitudes larger than for model-based techniques, as information is stored multi-redundantly, limiting either the size or resolution of an object or environment drastically. Second, image query and retrieval in real-time is a non-trivial, not yet solved problem. Third, the rendering and warping techniques are slow, since there is currently no hardware support, so that rendering has to be performed offline rather than interactively. One of the few attempts to implement a truly image-based rendering engine was the creation the PixelFlow supercomputer [McAllister et al., 1999] in order to obtain interactive frame rates, but there is currently no system for normal consumer PCs. Additionally, for many applications such as urban planning, simulations and games, it is inevitable to possess explicit information about the actual geometry of objects or environment in order to perform computations or add artificial components correctly. Only a geometric 3D model is capable of fulfilling these needs.

Furthermore, model based approaches have the advantage of being widely supported by various tools, hardware, and software standards. Starting from 1996, affordable hardware support for rendering textured triangular meshes on standard PCs has begun with the release of the first 3Dfx Voodoo chips set and 3D graphics card. Since then it has been advancing in increasingly shorter product cycles, mainly fueled by the gaming industry. While the first graphics cards had not more than 2 MB onboard memory, common gaming cards as of 2002 have 128 MB memory, massively parallel architectures, and various hardware-accelerated rendering features such as fog, realistic water surfaces, shading and many others, at rendering speeds of more than 100 million triangles per second. The operating principle for all 3D graphics cards is *z-buffering*, where for each pixel on the screen not only color but also an assigned depth z, i.e. distance from the viewing screen, is stored. If rendering of a new 3D triangle is requested, a pixel is only overwritten by the new triangle's pixel, if its depth is lower. Thus, occlusion is directly handled by hardware. OpenGL, originally developed by SGI, and DirectX, developed by Microsoft, provide convenient APIs to the graphics hardware. With these APIs, it is

simple to use the extensive hardware capabilities without detailed knowledge about the complex low-level processes. For example, triangles can simply be described by their vertex coordinates, translations or rotations by matrices, view points and light sources by position, and material properties by a set of intuitive parameters.

With the Virtual Reality Modeling Language (VRML), a powerful standard for describing and storing 3D models independently from the computer platform has been in existence since 1994. VRML offers not only the possibility of describing various geometry primitives and surface properties, but also scene graphs, handling of multiple levels of details, instantiation, diverse lighting sources, background scenery, and 4D animation. With the VRML97 standard, it also has an extensive interface to JAVA programs and functions for event handling. VRML models can be interactively viewed with web-based VRML-browsers such as Cosmo or Cortona Player, available at no costs as plug-ins for the standard web-browsers Netscape Navigator and Microsoft Internet Explorer. As a result, 3D models can easily be transferred to and viewed by a large audience. This is an increasingly important advantage for commercial applications such as Internet shopping, where customers prefer to inspect a product virtually before buying, and virtual walk-thrus, where a potential city visitor or hotel guest wants to explore the settings of the environment before booking.

Although hardware limits have been more and more extended, it is still a non-trivial problem to handle extremely large models with millions of triangles and many texture images, since such models can exceed the graphics memory by orders of magnitude. Typically, most parts of a large model are not visible from a given view point, or they occupy only a small space on the screen. Therefore, sophisticated techniques have been developed to budget the available graphics memory and select which parts of the model have to be loaded at which resolution. Typically, a complex model is structured into submodels, and during rendering a selection process called *culling* is applied. Popular methods are *frustum culling* to eliminate submodels not in the field of view, *distance culling* to cut off remote parts, and *visibility culling* to eliminate certainly occluded submodels. The more culling methods are combined, the less operations are for the graphics card to be performed. An equally important step, which has to be done during model generation, is creating multiple resolutions for each submodel called *levels-of-details* (LODs). As the name implies, lower LOD represent a given geometry by a smaller amounts of triangles, hence omitting more and more details while reducing mesh complexity. As an example, in its highest LOD, a house facade is represented by thousands of triangles and contains small features such as house entrances, windows and mailboxes, eventually even the brick structure. In its lowest LOD, however, a facade can be simply represented by a rectangle. The model structure and the hierarchy of submodel are defined in a *scene graph*, and the renderer uses this graph for both culling and determining which LOD to show under which conditions. Most commonly, the selection of the appropriate LOD depends on the distance of the object to the current viewpoint. These sophisticated techniques make systems like VGIS possible, [Ribarsky et al., 2002], which enables browsing Gigabytes of 3D terrain and urban data interactively.

2.2 Acquisition of 3D Models

Coincident with soaring rendering possibilities, there has been a continuously increasing demand for acquiring 3D models of existing objects and environments. A tremendous amount of work has been focusing on capturing small-size objects such as figures, statues, heads, and work pieces for virtual reality or reverse-engineering purposes. Such objects are captured in controlled laboratory environments, from known sensor positions, and under stable acquisition conditions. All sides can easily be accessed and the small size of those objects makes handling the acquisition simple. Acquiring detailed 3D models of buildings in outdoor environments is generally harder, since objects are large and immobile, and hence it is the sensors that have to be moved. Additionally, the environment cannot be controlled, since it is shared with people, and lighting conditions change dependent on time and sensing direction, since the light source is the sun. Aggravating is also the presence of many "uncooperative" materials such as glass and shiny steel, whose reflecting characteristic is problematic for most sensor types. Hence, the suitability of model acquisition techniques is different for objects and for environments, and in the following overview over existing methods we focus on applications to create 3D models of architectural structures.

The most basic and most time-consuming approach for obtaining a 3D model is to enter coordinates manually, e.g. from CAD drawings, construction plans, or survey measurements. This method is reliable and easy to apply without additional equipment, and large libraries of hand-made VRML models on the web witness its widespread use throughout the last years. Artificial objects or worlds are generally hand-made, for example for the large gaming market, and the demand for convenient editing and manipulation has lead to the development of an endless list of powerful 3D editor tools such as 3DstudioMax, AutoCAD, MultiGen Paradigm, Maya or Rhino. Using theses tools, modeling real-world scenery is the same process than creating an artificial model that resembles the real-world original. One of the early attempts to facilitate manual editing for large-scale architectural structures was suggested in [Bukowski and Séquin, 1995], in the context of the Soda-Walkthru project [Séquin et al, 1993]. In this project, indoor and outdoor appearance of a building was simulated before construction. Based on entering floor coordinates from construction plans, a 3D model was constructed and its consistency verified. For urban simulations, [Chan et al., 1998] manually created a large model of downtown Los Angeles using the MultiGen editor, based on plans from the urban planning department and by marking and extruding building footprints in aerial photos. In order to make the model appear photo-realistic, they manually texture-mapped the building tops with the aerial photos and the facades with pictures taken from ground level with a digital camera. Similarly, [Heller et. al., 1996] developed a system that facilitates the input of multiple types of data such as ground plans, terrain elevation maps, and aerial images. However, construction plans are neither always available nor always identical with the actual building, and manual editing is potentially error prone and time consuming, especially if a high amount of details is desired.

To overcome these drawbacks, automatic or semi-automatic approaches have been developed, using sensor data for the reconstruction process in order to achieve a higher level of automation. Additionally, sensor data allows capturing true shape, since resulting 3D models are based on real measurements of the actual geometry instead of idealistic plans. Cameras and laser scanners are by far the most common devices used for model reconstruction. While every method of obtaining photo-realistic models relies on camera images for texture acquisition, geometry can be obtained using either cameras or laser scanners, with the latter method becoming increasingly popular. In this case, there is an additional registration problem between laser-based geometry and camera-based texture, which does not occur if geometry is reconstructed from camera images alone.

2.2.1 Cameras and Stereo Vision

Geometry reconstruction using cameras is generally based on stereo vision. A camera performs a perspective projection of a 3D structure onto a 2D image, and each image pixel corresponds to a direction in 3D space, i.e. a ray originating in the camera's center of projection. Since all points along this ray project to the same pixel, it is not possible to reconstruct 3D coordinates given solely this one pixel. The principle of stereo vision is to use (at least) *two* cameras observing the scene from different viewpoints, to identify corresponding pixels and thus rays from their center of projection, and to compute a 3D vertex as rays' intersection, as shown in Figure 2-2. This principle is also known as triangulation, since the 3D vertex and the cameras' centers of projections form a triangle. The distance between the cameras is called *baseline*.

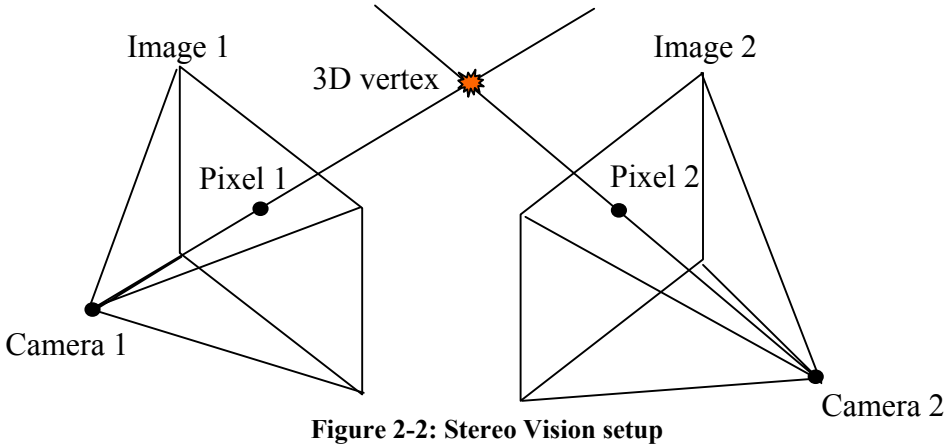


Figure 2-2: Stereo Vision setup

If the scene is static, it is possible to use only one single camera and take two images from different positions subsequently. In the plain stereo vision approach, capturing pose for both has to be known precisely; however, there are algorithms such as structure-from-motion which can estimate both geometry and acquisition poses for a series of images.

Although the idea of stereo vision is simple, there are two major problems in the practical implementation. The first problem is to exactly determine the direction in 3D space corresponding to a pixel, generally known as the *calibration problem*. For a given pixel, the direction does not only depend on the *extrinsic* camera parameters position and orientation, but also on *intrinsic* camera parameters such as lens distortion, image center and CCD chip pixel size, which are due to the non-idealistic nature of technical cameras. The internal parameters depend on the underlying camera model; in photogrammetry, the most accurate form of stereo vision, the camera models are complex and utilize many parameters. Calibration has been a well-studied problem during the last decades. For computer vision applications, relaxed camera models are used, modeling in particular the lens distortion as simply radial. Popular calibration methods are Tsai's camera calibration [Tsai, 1987], with five intrinsic parameters, and Zhang's "easy camera" calibration [Zhang, 2000], with six intrinsic parameters. Both methods are semi-automatic and use calibration objects with known geometry in order to compute the parameters. The obtained calibration accuracy is satisfactory for the vast majority of applications; however, once the camera is calibrated, its settings must remain unchanged. Besides these approaches, a new class of auto-calibration methods has been developed, which estimate the camera parameters during camera motion. These approaches have the advantage that no pre-calibration has to be performed, and as such, even sequences recorded with completely unknown parameters can be interpreted. Furthermore, camera parameter settings, in particular the focal length, are allowed to change during a sequence. The calibration accuracy, however, is lower than for pre-calibrated cameras.

The second, much harder problem is to identify which pixels correspond to each other in the two images, generally known as the *correspondence problem*. In semi-automated approaches, the pixel correspondences are usually defined by the user. For fully automated approaches, there are various algorithms as how to find correspondences. Typically, a first step is the usage of image processing algorithms to detect features such as corner points in both images. To find the correctly corresponding pairs, correlation can be applied to match the two images in a window around the feature points; this requires similar perspective and thus a small baseline, resulting in a high sensitivity to inaccurate feature location. There exist a variety of methods to reduce search space and hence accelerate the search process. For example, the correlation can be reduced to a 1D search, since the 3D location of a feature is along one line defined by the pixel in the first image, and hence all possible matching locations must also be along a line in the second image, called *epipolar line*. A *rectification* process can warp the images so that all epipolar lines are parallel to the horizontal image axis. Once the correct correspondences are found, the 3D location of the feature can be determined by intersecting the pixel rays.

The problem of finding correspondences is the most crucial part in stereo vision. If features are absent, for example on a white building wall, it is not possible to determine its shape with stereo vision methods. Hence, the spatial distribution of features depends highly on the scene and is typically non-uniform, resulting in large gaps with no 3D information. Problematic is also the opposite case, when too many features are present and it is not possible to identify and assign correspondences correctly, for example for a

wall with a repetitious brick pattern. In this case, stereo matching results are unpredictable, many obtained vertices are erroneous, and extracting geometric primitives is difficult. A significant improvement for such situations has been the usage of the RANSAC algorithm [Fischler and Bolles, 1981] to find geometric structures such as planes, yielding correct result even in presence of many outliers. Due to its robustness, it has become increasingly popular for computer vision applications during recent years.

2.2.2 Laser Scanners

The usage of laser light (Light Amplification by Stimulated Emission of Radiation) for distance measurement purposes has become extremely popular with the spread of the laser diode as an inexpensive source. With laser diodes, it is possible to create monochromatic light rays with extremely low beam divergence, switchable at Gigahertz speeds by electric signals. Dependent on the intended range, different properties of the laser light are exploited for distance measurement. Traditional interferometry achieves sub-nanometer resolution, and auto-focus methods achieve sub-micrometer resolution over millimeter range. For sub-meter and lower meter range, e.g. for scanning small objects such as figures or persons, triangulation is the most common measurement principle. Using the same idea as in stereo vision, a laser point or line sweeps over an object, while the scene and the backscattering light spot is captured with a calibrated camera, placed at a baseline distance apart from the laser. The object distance is determined in the triangle formed by the known baseline and the two base angles of camera and laser.

For far-range applications, starting from few meters up to many kilometers, the time-of-flight principle is used. Commonly, in reference to the well-known RADAR, this principle is also called LIDAR (Light Detection and Ranging), especially in the context of remote sensing. As illustrated in Figure 2-3, the time between sending a laser signal to an object and receiving its backscatter is measured, and the distance is computed over the known speed of light.

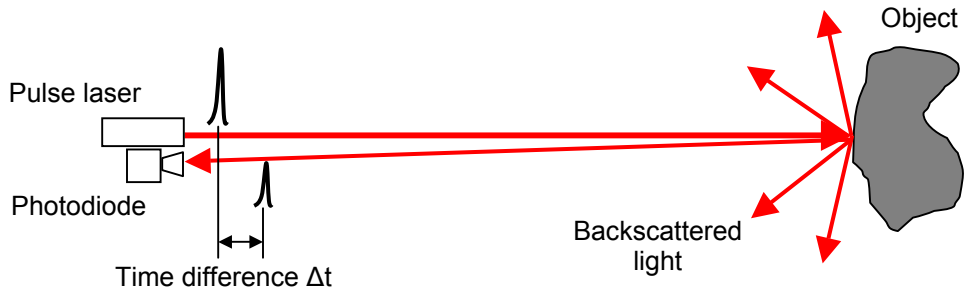


Figure 2-3: Distance measurement by time-of-flight

A laser scanner combines such a distance measurement unit with a deflection unit, in which the light paths are deflected by a rotating mirror and directed to various locations

on the object, as shown in Figure 2-4. The laser beam sweeps over the surface and scans the object. As the motion of the mirror is controlled and known precisely, a distance value and its corresponding mirror angle can be combined to compute a scan point, i.e. a vertex in the scanner's coordinate system. In a 2D scanner, the deflection unit contains only one rotating mirror; the laser beam sweeps along a line and all scan points are in a scanning plane orthogonal to the axis of rotation. In a 3D laser scanner, where the deflection unit contains either two rotating mirrors or one mirror with two degrees of freedom, the beam is deflected in two directions, and scan points are received in a row-column fashion. While the resulting scans are generally referred to as 2D scans and 3D scans, they are in fact rather 1½ D and 2½ D representations, since the laser light does not penetrate surfaces and is not able to capture any occluded structure.

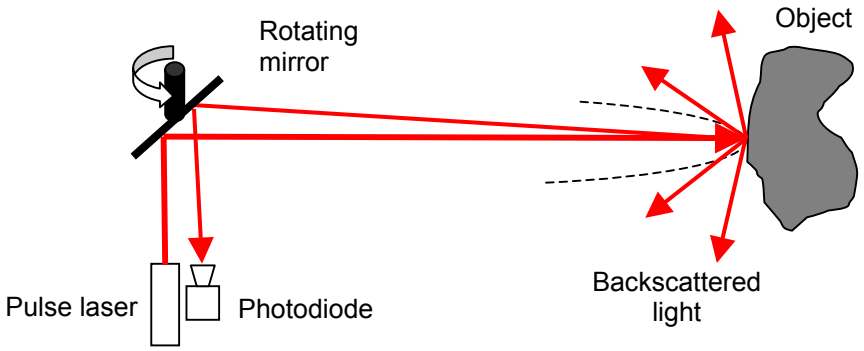


Figure 2-4: Time-of-flight measurement in a 2D laser scanner; both transmission and detection path are deflected by a rotating mirror.

Since the speed of light is high, the time measurement has to be extremely precise, requiring high bandwidth for transmitting laser diode, receiving photo diode, and the entire system. Determining the time of flight is either done directly by transmitting a light pulse and counting a clock signal until the pulse's reflection is received, or by amplitude-modulating the laser light and exploiting the phase shift between transmitted and received signal. Commercial scanners utilize sophisticated distance measurement devices, which consider various hardware signal response times to achieve time measurement accuracy in the sub-nanosecond range, thus equaling a distance accuracy of a few centimeters. The amount of backscattered light obviously depends on the surface. For example, the reflectivity of a black surface is only about 5% of the reflectivity of a white wall. It also depends with $1/r^2$ on the distance r to the object, since the backscattering (ideally) returns the light in the entire half sphere. Hence, the detection limit of the optical receiver limits the range of the distance measurement, in contrast to passive light sensors such as cameras. Additionally, the laser beam divergency is finite, yielding a light spot size increasing proportionally to the distance. Beam divergency can be substantial and depends on both the desired laser safety class and the intended application. For security applications, it is desired that there are no gaps between subsequent scan points, so that the scanner detects every approaching object. For measurement applications, usually a small spot size is desired in order to avoid an averaging effect over a large surface area. The SICK LMS scanners that we use in our approach were originally mainly developed

for security applications; their relatively large beam divergency of about 15 milliradians results in a spot diameter of 1.2 meter at the maximum range distance of 80 meters. If there are multiple distances within the laser spot, e.g. if the spot hits two object at slightly different distances or a wall under an oblique angle, the shape of the backscattered pulse is a mixture. In this case, the resulting distance value is potentially inaccurate or erroneous.

In the following sections, we will give an overview over existing 3D modeling approaches based on camera images and/or laser scanners. For this overview, it is reasonable to distinguish between ground-based and airborne data acquisition and modeling strategies, since the two scenarios differ substantially in image perspective, resolution, occlusion problems, and amount of acquired data.

2.2.3 Model Generation from Airborne View

The use of remote sensing data, i.e. aerial images or airborne laser scans, has been investigated in several 3D modeling approaches. Since aerial images are widely available, building reconstruction based on stereo matching has been of high interest in the last decade for both civilian and military purposes. Examples for automated and semi-automated approaches for building reconstruction can be found in [Gruen et al., 1995] [Frere et al. 1998], [Huertas et al., 1999], [Baillard and Zisserman, 1999], [Förstner, 1999], [Kim et al, 2001, [Brenner et al., 2001], and [Vestri and Devernay, 2001]. The 3D modeling process can be distinguished into building detection, structuring, and reconstruction steps.

Most difficult are the first two steps: detecting a building and determining its outline and structure. Once the dimensions are known, the geometric reconstruction based on stereo matching is rather straightforward, e.g. it can be computed automatically by image correlation with a window according to the building shape. Completely automated approaches attempt to extract building shapes in a intensity-based segmentation process, and most commonly, a priori knowledge about building shape is assumed, e.g. by making strong restrictions about possible architectural structures. Despite these efforts, completely automated building extraction based on images has not yet led to acceptable models, and reported success rates generally refer to very specific conditions. It has been noted by various authors that fully automated approaches cannot be expected to deliver completely satisfying results in the near future, due to the enormous variety and complexity of architectural structures.

Semi-automated approaches require manual steps by an expert user, e.g. the selection of feature points or building outlines. Typically a model resolution in the range of one foot to one meter in the horizontal directions can be achieved from airborne view, whereas the vertical precision depends on the baseline of the images and is generally lower. These numbers, however, do not tell the entire truth: only the objects selected in the labor-intensive manual or error-prone automatic structuring process appear in the final model.

In semi-automated approaches, objects are typically simply omitted due to the limited labor time. In automated approaches, however, it can also occur that actually non-existing structures appear in the model, e.g. if the segmentation process is too "generous" in declaring image areas as buildings. Hence, the difference between real world and model can be quite substantial.

An alternative is the usage of airborne laser scans, which has become increasingly popular in recent years, due to enormous advances in technology. A city is scanned from airborne view with a far-range 2D scanner mounted on a plane, and from the obtained scan points a Digital Surface Model, i.e. an array of regularly spaced altitude samples, is created. We give an overview over previous work in Chapter 7, where we also suggest the usage of airborne scans to complete ground-based models.

2.2.4 Model Generation from Ground-Based View

The data acquisition from airborne view can obviously neither capture detailed facade geometry nor facade texture, hence making models only suitable for virtual fly-thrus, not for walk- or drive-thrus. Consequently, there has been a variety of approaches that attempt to acquire models from the ground-based perspective familiar to humans. For indoor environments, ground-based modeling includes capturing room interior as well as walls and ceilings. For outdoor environments, ground-based modeling mainly equals modeling the facades of buildings, and thus ground-based models are almost completely complementary to airborne models.

Capturing ground-based data of large areas requires the acquisition system to be moved within the environment, and the used sensor platform inherently affects speed and accessible area for the capturing process. It is also generally necessary to determine the position of the devices either directly during data acquisition or in a postprocessing step. The most flexibility is offered if devices are handheld and can be moved freely in all six degrees-of-freedom, but in this case pose tracking is difficult since no motion restrictions apply. Many sophisticated acquisition systems are too heavy to be carried by a human over longer distances, and therefore in most previous work, capturing devices have been mounted on wheel platforms such as trolleys, cars, and robots, inherently constraining motion mainly to the degrees of freedom given by the ground surface. The usage of mobile robots as acquisition platforms is quite common, since mapping and localization are key issues in robotics. One of the advantages of robots is that the exploration of dangerous environments, e.g. mine fields or hostile areas, can be done remotely without putting humans at risk. It has been envisioned that using autonomous rather than remote controlled mobile robots could automate not only the model reconstruction, but also the entire data acquisition process. However, while the general idea is striking, practical implementation has proven to be difficult. Particularly outdoor environments are extremely complex, and mobile robots are far from being reliable enough to explore an entire city autonomously. Additionally, mobile robots are slow in comparison to cars, which are capable of traversing even large cities entirely within few hours.

Similar to the airborne model construction, there have been popular semi-automated approaches to image-based modeling. The most prominent one is Debevec's "Facade" approach [Debevec et al., 1996], which is the source of the commercial software packages Canoma and ImageModeler, and has been utilized to create the photo-realistic building models for the "Campanile Movie" and "The Matrix Reloaded". Only a small set of images taken from different viewpoints is necessary to construct a model. For each object to model, the user first defines the underlying simple geometric shape such as a box or pyramid, and marks the corresponding edges in the images. Then, the algorithm determines the actual dimensions by minimizing the edge deviation in the images, resulting in a polygonal representation of the geometry. As an additional byproduct the camera positions are obtained, and during rendering the texture is taken from the image most similar to the viewing perspective. Due to this view dependent texture mapping, Debevec's work is often referenced as an example for image-based rendering, although it is in fact rather a hybrid approach. As demonstrated in the "Campanile Movie", it is possible to obtain surprisingly realistic looking results with this technique, though at the cost of substantial manual interaction. A variation of this idea towards more automatism can be found in [Cipolla et al., 1999]**,** making more restrictive assumptions about architectural structures such as orthogonality and parallelism. Similarly, [Dick et al., 2001] and [Werner and Zisserman, 2002] propose extensions using additional cues such as vanishing points and texture to find correspondences for wide-baseline stereo matching. Using robust RANSAC methods, planar patches are extracted from the unorganized and partially erroneous point cloud obtained from stereo matching. The given examples, however, refer to single buildings with rather simple structure and clearly visible vanishing lines, and neither occluding foreground objects nor extended glass windows were present in the scenes.

Other more automatic approaches use video streams as dense series of images and structure-from-motion to recover geometry. [Faugeras et al., 1998] and [Zisserman et al., 1999] were able to track features over multiple frames and managed to recover 3D points and lines based on this multi-view stereo vision. To construct a polygonal model from the sparse 3D vertices is by no means trivial, and Faugeras' approach required some manual intervention to fit planes to the 3D vertices. Using RANSAC, Zisserman extracted planes automatically, though again for a scene of low complexity. As Faugeras states, neither theory nor technology are ready yet for fully automated modeling from image sequences.

Besides the issues of detecting and correctly identifying features across multiple images, the resolution and the noise sensitivity for determining 2D features and hence 3D vertices pose a hard problem for image-based approaches. Especially for video data, recovered poses show a drift-like behavior over multiple frames due to the low resolution, and closing loops does not lead to consistent models, unless cross-image verification is performed. This comes usually at the cost of $O(n^2)$ computation complexity with the number n of frames, and imposes hence severe scaling limits for the length of and image sequence. An innovative approach to a more accurate image-based reconstructed has been developed in the MIT city scanning project [Teller, 1998], [Antone and Teller,

2000]. A cart, equipped with a high-resolution camera on a pan-tilt unit and various positioning devices is moved to several locations. At each, the camera tilts and pans, taking multiple images from the entire half sphere. In a mosaicing process, these images are combined to a spherical image, thus creating an artificial fish-eye view with an unprecedented resolution of tens of megapixels per image. Since this image has a maximal possible effective field of view at a high resolution, it is possible to estimate features and vanishing lines accurately, enabling to estimate pose in respect to other spherical images precisely and create a network of mutually referenced images. However, while camera pose estimation is comprehensive, this approach has not yet solved essential general drawbacks of image-based model reconstruction such as the irregular density of 3D samples and the problem of creating a model from those samples. Also, acquisition speed is limited since data has to be collected in a stop-and-go fashion, resulting in several days acquisition time for an entire city.

[Kawasaki et al., 1999] suggest an interesting approach, which uses video only for texturing an already existing digital model, e.g. obtained from semi-automatic airborne stereo matching. While driving in a city with a car, a video stream is captured and processed offline, in order to track the vehicle and fit the video images as texture on the model facades. Building outlines are found as vertical cuts, and block matching is used to estimate velocity. A coarse depth map of objects in the video stream, which is at least sufficient to discriminate intersections, is obtained by exploiting parallax effects. With this information extracted, the video stream can be matched with the model. This approach is remarkable since it is car-based and as such potentially capable of covering an entire city area in acceptable time. However, a pre-existing geometric model is required, and the authors note that foreground objects cause errors in both the matching process and the texture usage. Additionally, it can be expected that the coarse geometry derived from airborne view differs substantially from the actually visible geometry at the ground level.

An alternative is to acquire this ground-based geometry by laser scanners. Three-dimensional vertices acquired by laser scanners are more reliable, accurate and dense than those from stereo vision, and in addition, measurements usually come in a regular grid shape. In previous work, 3D laser scanners have been used, providing directly a grid of 3D vertices from a single view. Typically, however, one single view is not sufficient for reconstruction, especially not for large-scale objects such as a building. If multiple views have to be combined to reconstruct an object, the regular grid structure inherent to the scanner is not preserved. Nevertheless, the density of 3D information is more uniform than for stereo vision and contains far less outliers, thus facilitating surface extraction significantly. While some scanners provide a reflectivity value for each scan point, they do not provide color or texture information, thus making the additional use of a camera for creating photo-realistic models inevitable. Hence, the additional problem of registering camera images and geometry has to be solved. In a nutshell, the typically addressed problems in laser-based model reconstruction are registration and merging of multiple views to one consistent representation, the subsequent reconstruction of a polygonal model, and finally the registration with camera images and texture mapping.

Merging the scans is straightforward if the pose of the scanner is known precisely for all views. However, in most scenarios is at best a coarse approximation of the actual pose available, and in this case a registration of the scans is necessary. Supposed the correspondences between the scan points of two 3D laser scans are known, it is fairly simple to compute the transformation parameters, for example by minimizing a square distance function. The problem, however, is to obtain the correct correspondences, preferably in an automated way. ICP (Iterative closest point, [Besl and McKay, 1992]) is probably the most prominent approach for automatically matching and registering two 3D laser scans, under the preliminary that acquisition poses are a priori approximately known. The ICP algorithm can be applied directly on the scans, without the need of extracting features such as lines. Iteratively, a guess for preliminary scan point correspondences is made by using a transformation independent rule, and then, based on these intermediate correspondences, the relative transformation is computed and applied. Besl and McKay suggested the *closest-point* rule, i.e. each scan point gets assigned its closest point in the other scan. They could prove that the ICP algorithm always converges monotonically to a local minimum, although it converges slowly in its plain implementation. ICP and its variations or similar matching methods have been used in many approaches [Pulli et al., 1997], [Yang and Allen, 1998], [Dorai et al., 1998], and a good overview is given in [Rusinkiewicz and Levoy, 2001].

Complete systems that acquire both 3D range scans and camera images for indoor environments can be found in [Sequira et al., 1996], [El-Hakim et al., 1998], [McAllister et al., 1999], and [Yu et al., 2001]. In the first three approaches, a camera and a laser scanner are mounted close to each other, so that both perspective and field of view are almost identical. Sequeira et al. provide an early example for a robot-based system, in which the entire chain from automated acquisition planning to registration, processing and geometric model reconstruction is addressed, though obtained model quality is low. El-Hakim et al. use a triangulation-based laser scanner and a camera, both registered by selecting control points of an accurate large-scale calibration object. Multiple views are registered by using dead reckoning as initial estimate and bundle adjustment for refinement. McAllister et al. calibrate the camera by correlating the camera image with the simultaneously captured laser reflectivity image; the registration of different views is done manually. They have also proposed an image-based rendering scheme, so that the typical polygonalization step is not required. Yu et al. placed reflective markers on the walls of a large furnished room; laser scans and photos are taken independently from each other, and automatically registered by detecting the markers in both modalities. Due to the high scan point precision and image resolution, their approach offers stunning models and furthermore the possibility of automatically segmenting and extracting objects via a normalized-cut framework. However, acquisition time is exorbitant and placing reference markers is not applicable to large-scale outdoor environments.

[Stamos and Allen, 2002] have equipped a mobile robot with a Cyrax 3D laser scanner and a camera in order to acquire building facades. They extract lines as intersection of segmented planar patches from the scans, and use these features to register range scans

with each other, and with vanishing lines from camera images. A set of solid volumes is created by sweeping multiple range views of a building, and the final geometric model is reconstructed as their volumetric intersection. Their approach is automatic and the resulting building model appears photo-realistic. However, due to the slow 3D scanner, acquisition time for a single building is already more than one hour, and hence their algorithm does not scale to more than a few buildings, so that it is not suitable for modeling an entire city. Furthermore, reliability issues apply for the complete autonomous data acquisition with a robot, which the authors have suggested as future work.

The reason for the more common usage of 3D laser scanners versus 2D scanners is that they conveniently provide measurements over an entire frustum in a common coordinate system, rather than only along one line; thus, each scan by itself is already a comprehensive piece of information. Typically, 3D laser scanners are designed as high precision measurement devices, and the high accuracy and density of 3D scan points enable relatively precise automated feature extraction. Since the 3D scan matching algorithms described above make it possible to register multiple scans and to determine their relative 3D pose accurately, requirements for initial pose estimation accuracy are rather low. This comes at the cost of a long acquisition time for an entire 3D scan, and since the system has to stand still during this time, a slow stop-and-go fashion for the data acquisition is required. Fast 2D scanners do not have this disadvantage, since acquisition time for a 2D scan is orders of magnitude smaller. However, 3D matching is not applicable to a set of given 2D scans in 3D space, and hence the scans can only be correctly merged if the pose accuracy of the acquisition system is extremely high. Since most previous approaches failed to provide this accuracy, they were unable to use the time advantage that 2D scanners provide. To our knowledge, there are only two systems that can acquire geometry with vertically mounted 2D scanners during motion, and are thus similar to our approach. One is described in [Thrun et al., 2000] and [Hähnel et al., 2001], where a Sick scanner is mounted on a mobile robot. It is aimed at creating and updating a geometrical 3D map of the robot's indoor or outdoor environment in real time, using the robot's own computational power. The real-time modeling capability is remarkable and important in the given context of robot map building. However, while the resulting models are suitable for robot localization and sufficient to provide an impression for the environment's geometry, the quality of both geometry and texture is clearly not acceptable if photo-realism is required. And again, the usage of a mobile robot limits the possible scale of the models. The other approach [Zhao and Shibasaki, 1999] is car-based and hence capable of traversing large-scale city environment in acceptable time. Geometry is captured using vertical IBEO laser scanners, and texture using a vertical line camera. A localization unit containing GPS, INS and odometry sensors is used to determine the vehicle's position during data acquisition. Recently, concurrently to us, Zhao and Shibasaki combined the vertical scanners with a horizontal 2D scanner [Zhao and Shibasaki, 2001], in order to meet the high position accuracy requirements for correctly merging vertical 2D scans. Assuming flat environments, they are able to track the vehicle based on scan matching. Since they do not have a global correction or accurate registration in respect to a global map, position discrepancies occur while

closing loops, and they do not consider a fusion with airborne models. The scope of their work is approximately Chapter 4 of this thesis, with comparable results.

As a summary of the above approaches, we can note the following:

- The vast majority of existing approaches is not applicable to the scale to an entire city, due to slow acquisition platforms, insufficient global registration, and exploding computation times for larger models.
- Purely image-based modeling approaches are currently and in near future not capable of reconstructing the various geometric structures occurring in a city in a fully automated way.
- Approaches using combinations of 3D laser scanners and cameras have been successfully applied for automated model generation, though requiring a slow stop-and-go data acquisition, and only for indoor environments or limited outdoor environments at the scale of one building.
- For combinations of 2D laser scanners and cameras, accurate localization in a large-scale outdoor environment is a crucial problem and has so far prohibited the use of these devices.

3 Ground Based Model Acquisition

In this chapter, we propose a new experimental setup that is capable of rapidly acquiring geometry and texture data of entire city areas at ground level. Data is acquired continuously, rather than in a stop-and-go fashion, and is subsequently processed offline. We further describe the implementation of an acquisition system that we have developed based on this approach, mounted on a truck moving at normal speeds on public roads.

3.1 Drive-by Scanning - A New Acquisition Approach

As summarized in the previous chapter, purely image-based approaches tend to fail in complex city areas, where occluding foreground objects, mirroring glass surfaces and changing lighting conditions are common. At the same time, the price of commercial 3D laser scanners, which are capable of providing true geometric measurements of the facades, is extremely high, thus rendering them unaffordable for many applications. Furthermore, a common disadvantage of existing ground-based city-modeling approaches is that the acquisition time for an entire city is unacceptably long, since data is collected in a stop-and-go fashion, and most acquisition platforms are not capable of querying large city districts rapidly.

In order to overcome these limitations, we propose “Drive-by Scanning” as a different approach to acquire 3D geometry and texture data of entire streets at high speeds. The principal idea is to scan building facades continuously while passing by, using 2D laser scanners instead of slow 3D laser scanners. Our experimental setup consists of two fast 2D laser scanners for geometry acquisition and a digital camera for texture acquisition, both mounted on a rack on top of an automobile. As shown in Figure 3-1, both 2D scanners are facing the same side of the street; one is mounted vertically with the scanning plane orthogonal to the driving direction, the other one is mounted horizontally with the scanning plane parallel to the ground.

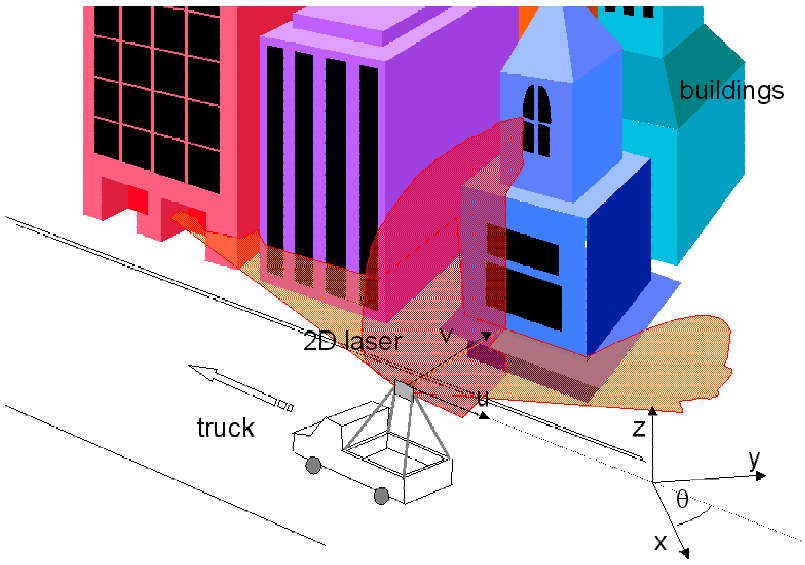


Figure 3-1: System setup

The devices are used as follows:

a) During vehicle motion, the **vertical 2D laser** scanner sweeps over the complete building facades and captures their shape. This setup is similar to the handheld color scanners that were common to scan documents in the early days of home computing, before flatbed scanners became inexpensive and popular. One could even regard this setup as an imaginary 3D scanner, for which the motion of the vehicle performs the task of the tilt mirror. However, under normal traffic conditions the vehicle's motion is by no means uniform, and hence the scanning planes are neither parallel nor equally spaced.

b) The **horizontal 2D laser** scanner measures the shape of the environment in a plane parallel to the ground. Subsequent scans are therefore taken in the same plane and overlap significantly. This scanner is used for the position estimation discussed in the next chapter, and enables pose accuracy in the sub-centimeter range.

c) The **digital camera** is used for capturing the texture of the facades. It is mounted with its viewing direction along the intersection of the two scanning planes, so that it faces directly the building facade.

d) The **automobile**, for example a pickup truck or a van, carries the sensors during data acquisition and is the power supply for the entire system. The sensors are mounted on top of a rack, so that obstacles such as cars and pedestrians in the field of view can be overlooked and occlusion is reduced.

This setup has several advantages over previous approaches: First, since the facades are captured while driving by, data is acquired in a fast continuous fashion rather than in a stop-and-go mode. Second, an automobile as an acquisition platform can be driven on public roads and under normal traffic conditions. With this mobile platform, it is possible to traverse an entire city in a few hours completely. Third, this setup is cheap; the enormous advantages of laser scanners for the geometry capture can be exploited without

an expensive 3D scanner. As of 2002, full 3D laser scanner usable for city environments, for example LMS from Riegl, Austria or Cyrix from Cyra, California cost 80,000$ and 200,000$, respectively. Additionally, if a differential GPS is used for determining global position during acquisition, the costs increase by 10,000$ to 20,000$, although the accuracy in city environment can be poor due to multi-path reflections. In contrast, two 2D laser scanners cost only around 15,000$ total, and are thus an orders of magnitude less expensive.

The usage of a car enables high-speed data acquisition; however, it also results in some limitations: this particular vehicle has to use roads, so that areas only accessible via narrow passages, e.g. backyards or buildings along trails, cannot be captured. However, neither sensor setup and nor the processing algorithms described in the next chapters depend on a car as platform. If accessibility is crucial, different vehicles can be chosen; e.g. a wheel chair, a boat or a helicopter could in principle be used as well. If a pickup truck or one of the other mentioned platforms is used, the data acquisition is not completely autonomous, since there is one manual step, i.e. driving. However, non-experts can perform this task, and we consider the duration of a few hours for an entire city not as prohibitive. The reliability and durability of cars is extremely high, whereas currently this is in complex city environment not the case for completely autonomous vehicles.

This particular method for data acquisition yields to some particular problems: First, rather than the 3D scan registration problem commonly associated with 3D scanners, we face a position estimation problem: In order to construct an accurate 3D model, the pose, i.e. the position and orientation of the truck and its sensor unit, needs to be determined precisely, and we propose suitable methods using the horizontal laser scanner in Chapters 4 and 5. Second, it is necessary to synchronize the two laser scanners and the camera accurately. Synchronization is not a critical issue for slow-moving robots, but a car can drive up to 25 mph in cities and hence an order of magnitude faster. Third, occlusion effects are significantly more problematic. An approach that registers and merges 3D laser scans typically results in a good scan point coverage for most surfaces, since individual 3D scans are taken from different viewpoints, and hence holes caused by occluding foreground objects in one scan are filled with scan points from another view. In our case, we scan the facades only once and from direct view, and since each foreground object blocks the laser beam, there is not information about the structure behind. This causes large holes in the facades, and we suggest algorithms to fill these holes by making a reasonable guess for the geometry in Chapter 5. Fourth, the size of the data stream during acquisition is enormous. Since an entire city can be captured within short time, the data throughput is substantial and data has to be stored rapidly.

3.2 Data Acquisition System

According to the ideas described in the previous section, we have developed a data acquisition system and have mounted this system on top of a pickup truck, as shown in

Figure 3-2. This system can be divided into two parts: a *sensor unit* and a *processing unit*. In order to avoid dynamic obstacles such as cars and pedestrians in the direct view, we mount the sensor unit on a rack, so that it is at a height of approximately 3.6 meters, the maximum the California traffic regulations allow. The processing unit consisting of a dual processor PC, large hard disk drives, and additional electronics for power supply and signal shaping is mounted in the truck bed.

Figure 3-2: Data acquisition system

As shown in Figure 3-3, the sensor unit consists of the two 2D laser scanners mounted with their scanning planes at exactly 90 degrees and a digital camera; these devices remain permanently fixed in respect to each other throughout the data acquisition.

Figure 3-3: Sensor unit

We use the LMS 291 laser scanner manufactured by Sick AG, Germany; this scanner has a 180^0 field of view with a resolution of 1^0, a range of 80 meters and an accuracy of ±3.5 centimeters. The acquisition time for a 2D scan sweep is 6.7 milliseconds, and the scan frequency is 75 Hz. If configured as master device, the scanner generates a 75 Hz signal corresponding to its scanning operation; if configured as slave device, it synchronizes its scanning operation to an external 75 Hz clock, and we use this feature to synchronize the two scanners. The connection to the processing unit is established via a 500 kbit/sec serial interface.

The camera for texture acquisition is also directed towards the side of the streets, with its line of sight parallel to the intersection between the orthogonal scanning planes, hence facing the facades. We use DWF-X 700 color camera from Sony corp., Japan, in combination with a 3.6 mm wide-angle lens. In this configuration, the camera has horizontally a 96-degree and vertically a 70-degree angle of view. The image resolution is 1024 by 768 pixels, and since the camera provides an image in the YUV (4:2:2) color system, a single image has a size of 1024 x 768 x 2 bytes = 1.5 Mbytes. The camera has automatic white balancing and shutter time. Each single image acquisition can be triggered, up to a maximum frame rate of 15 Hz. The camera is linked with a fire wire connection and a trigger line to the processing unit. As a one-time task before the first data acquisition, we calibrate the intrinsic camera parameters using Zhang's "easy camera" calibration [Zhang, 2000], which has the advantage that it models the substantial radial distortion of the used wide-angle lens with two intrinsic parameters. Furthermore, we determine the remaining extrinsic camera parameters: due to construction, the relative position of the camera in regard to the scanners is known precisely and the orientation angles are accurate to a few degrees; hence, the rotations are decoupled and it is simple to fine-tune orientation parameters manually.

The processing unit shown in Figure 3-4 is mounted in the bed of the truck, on damping material in order to reduce mechanical shocks. Its core is a PC with two 850 MHz Pentium-III processors running Windows 2000, and it has a counter/timer board, a high speed serial interface card, an IEEE 1394 fire wire connection and a RAID hard disk drive array. Additionally, the unit contains electronics for power supply, signal shaping, and trigger signal generation. The key feature of our data acquisition system is that all devices are accurately synchronized; this enables us to automatically determine correspondences between the laser scanners themselves and the camera. The horizontal laser scanner is configured as master scanner; its 75 Hz signal is the time base of the data acquisition. The vertical scanner is configured as slave, and synchronizes itself to the master's time base. The counter/timer board counts the master's clock pulses in order to generate time stamps for the data, and it divides the clock to obtain a trigger signal for the camera.

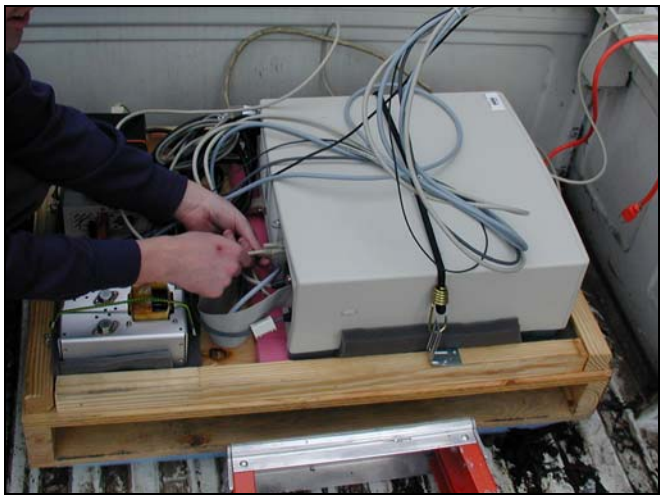


Figure 3-4: Processing unit

Since the laser scanners and the camera deliver data without waiting for acknowledge signals, this data has to be picked up immediately or is lost otherwise. This is especially critical for the scanners, which communicate via a 500 kBaud serial interface: in the scanner's data protocol, the position of a distance value in the byte stream determines the angle under which the measurement was taken. If only one single byte is dropped, the correct angle cannot be determined any more. Even more severe, the synchronization of the data communication is lost entirely, since the control software expected a certain amount of bytes to be read in a package, and hence waits until finally the first bytes of the next scanning cycle have arrived. These bytes are then missing during the readout of the next cycle, and so on. Therefore, it has to be ensured that no byte of the laser data stream is dropped at any time.

We have developed a data pipeline and extendable multithread software to capture and store the incoming data streams, as schematically shown in Figure 3-5. Although the operating system Windows 2000 is technically not a real time operating system, it can be brought to sufficient real time behavior for our application if system services such as networking are turned off, and the process priority of the acquisition threads is set higher than the priority of any operating system process. For each input device, we run an associated *device reader thread* that waits for incoming data and stores it in a common ring buffer memory. A *writer thread* with a lower priority than the readers continuously writes down the ring buffer to the RAID hard disk drive array and frees the corresponding ring buffer section subsequently. Both scanners together generate a 1Mbit/sec data stream, whereas the data stream generated by the camera at full frame rate is with 15 Hz x 1.5 Mbytes = 180 Mbit/sec two orders of magnitude larger. Since currently the writing capacity is with an average of about 100 Mbit/sec the bottleneck of the data acquisition, the camera cannot be operated at its maximum frame rate. We choose to divide the master scanner's clock by 15, so that a frame rate of 5 Hz results. Since the speed limit in cities is 25 mph, the spacing between subsequent laser scans is

then at worst 25 mph/75 Hz = 14.8 cm, and the spacing between subsequent camera images is not more than 25 mph/5 Hz = 2.22 m for this configuration. To assess the maximum rotation between subsequent scans, we can e.g. assume that while driving, the maximum angular velocity is never more than 45 degree/sec, i.e. the vehicle does not perform a full 90 degree turn faster than in two seconds. Then, the rotation between subsequent scans is maximal 45 degree/sec/75 Hz = 0.6 degree, and between images 45 degree/sec/5 Hz = 9 degree.

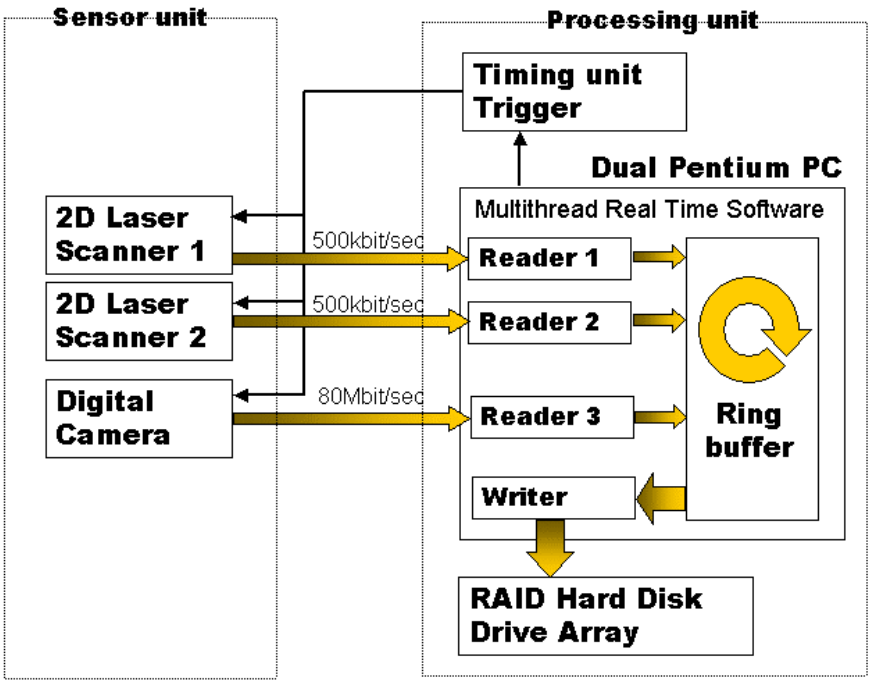


Figure 3-5: Schematics of data acquisition system

During data acquisition, the system is entirely busy with capturing data and storing it on the hard disk drives; there are neither time nor resources for further processing. The raw data, a memory dump of the counter/timer board registers and other system parameters are directly written to one single continuous file, since creating new files or directory decreases writing throughput of the hard disk drives. After the acquisition is finished, data conversion is necessary as the first post-processing step: the memory dump of the hardware registers enables us to identify corresponding scans and camera images, and we separate the raw streaming file into its content components, and assign each scan and image a correct time stamp. Additionally, we use Microsoft's EasyCamera software and the internal parameters from the pre-calibration step in order to remove the distortion from the camera images. After these steps, the data is in a form suitable for the actual postprocessing and model generation, which is described in the next chapters.

4 Tracking the Acquisition System

In this Chapter, we describe our approach to determine in an offline computation the sensor unit's pose in the city environment during data acquisition. Laser scans and camera images are given in the local sensor coordinate system, and an individual scan or image is of no use unless it is combined with subsequent data sets to a coherent model. Therefore, it is necessary to know the relative pose between the acquisition instants. If the model has to be globally correct, it is not sufficient to know the relative pose only; rather, the absolute global pose in respect to a world coordinate system has to be known accurately for each data set. All devices are fixed on the sensor unit and mounted onto the vehicle, and hence we use in the following generally the term localizing the vehicle when we strictly speaking mean determining the pose of its sensor unit. In this Chapter, we describe how to obtain relative 2D pose estimate and an initial path for the vehicle by matching subsequent horizontal laser scans. This initial path is the base of a global localization scheme to be detailed in the next Chapter.

Determining one's position and orientation is one of the oldest problems of mankind, and was already a critical issue in ancient times. While first sailors used magnetic stones or stars as guides for the orientation, today's modern equipment includes advanced tools such as optical gyroscopes and Global Positioning System (GPS). One can make a distinction between devices that deliver pose estimates absolute, i.e. directly in respect to one global coordinate system, or relative, i.e. the pose change between two events is obtained in respect to their local coordinate system. While it is convenient that first ones provide the global pose directly, the resulting error for computing the relative pose, i.e. the difference between two subsequent global poses, can be as large as twice the maximum system error; this can be a multiple of the actual motion. For the second type of devices the obtained relative estimates between subsequent positions are often highly accurate, while the absolute pose can only be computed by concatenating the relative estimates, or steps, to a continuous trajectory in the global coordinate system. According to robotics, we refer to this relative positioning as *tracking*. Since errors in the relative estimates are inevitable, there is an error in global position increasing over time, and once a severe error is introduced, it persists throughout the entire following trajectory, if there is no global correction available.

To illustrate the difference between relative and absolute pose estimates, one can compare two methods of ship navigation: When Columbus traversed the Atlantic to discover America, he obtained an estimate for the orientation from a magnetic compass, and an estimate for his current speed by throwing a piece of flotsam over the side of the ship and counting the time it needed to pass two marks. Then he used dead reckoning to determine his position: Starting from a know global position, he iteratively calculated the traveled distance as time multiplied by speed, and added this distance according to the ship's orientation to the last position in order to obtain the next one. In contrast, a modern GPS system provides global position estimates e.g. at 10 meter accuracy. Since the dead reckoning approach can result in global position errors of hundreds of miles after long travels, GPS is more accurate in global position terms. However, if an estimate for a

small relative changes is requested, e.g. between two close positions few centimeters apart from each other, subtracting the corresponding GPS readouts would result in a large error due to noise, and in this case dead reckoning provides a more accurate estimate. In the case of a ship, it is possible to overcome the noise problem by assuming inertia and smoothing GPS readouts accordingly. However, this assumption cannot always be made: one imagine for example a rabbit running in a field: the exact details of its zickzack trajectory cannot be determined with a GPS, and localizing a vehicle moving in a city, where traffic situations require sudden changes in direction, is a similar case. In this and the following chapter, we will present a more adequate localization approach for this situation.

To avoid confusion, we use the term *pose* throughout this thesis for the combination of position and orientation in a given Cartesian coordinate system. *Position* is the location in space described by three coordinates, e.g. x, y, and z in a 3D Cartesian coordinate system. *Orientation* is the attitude of an object in respect to the coordinate system, described by three angles yaw, pitch, and roll as common in avionics and shown in Figure 4-1. Therefore, pose has six degrees of freedom (DOF) and is also referred to as *3D pose*. If we refer to a *2D pose*, we mean the pose for an object constrained to motion within a plane, which can hence be entirely described by the two position parameters x and y and the orientation angle yaw. *Global pose* is the pose in respect to a geographic coordinate system. For the dimensions of a city, we can neglect the effect of earth curvature and define a global Cartesian coordinate system with x- and y-axis horizontal, and the z-axis pointing to the sky. For example, according to the UTM NAD83 and NAVD88 standards, x could be easting, y northing, and z the altitude of a position.

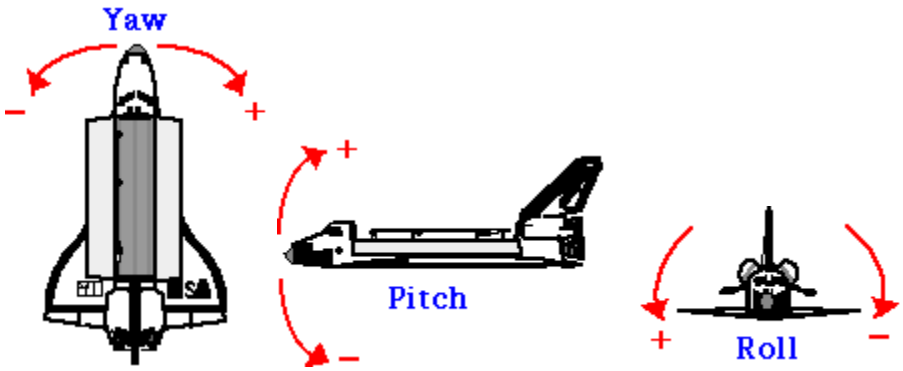


Figure 4-1: Attitude described by yaw, pitch, and roll angle. (*Source: NASA*)

4.1 Relative Pose Estimates

Relative pose estimates for a vehicle can be obtained by a variety of techniques. One of the oldest approaches for obtaining an estimate for a vehicle's motion between two times is dead reckoning based on odometry, in which the rotations of the individual wheels are counted, eventually in combination with the steering angle, and converted to a relative

pose by using a kinematical model of the vehicle. The kinematical model is an idealistic abstraction of the vehicle, and effects such as low tire pressure or slippage on the ground are not considered. Also, uneven terrain or bumps cause discrepancy between idealistic and actual motion, hence generally reducing accuracy. Other kinematical approaches determine relative displacement by integrating speed or double integrating acceleration, respectively: Speed can be measured with sensors such as a radar velocimeters, in which the Doppler effect is used to create a mixed frequency proportional to the speed, and inertial sensors, highly integrated with MEMS technology and used e.g. in airbags, provide an estimate for the acceleration. However, due to the necessary integration, even small offsets in speed or acceleration result easily in large drift errors in position, and therefore theses sensors are for only usable in combination with slower, more accurate devices. To determine orientation changes, optical gyroscopes can be used; these are expensive high-end devices, based on interferometry and exploiting the quantum-mechanical Sagnac effect to determine rotation speed. While these devices display an impressive precision and are e.g. used in avionics, they come at exorbitant costs.

To reduce shortcomings and errors of each individual localization approach or device, they are often combined in a data fusion process. Multiple, potentially conflicting sensor readings are combined to a single state estimate, for example by using a Kalman filter [Kalman, 1960], which takes into account both an uncertainty measure for the current state variables and the time-dependent credibility of the individual sensors.

Due to the enormous advances in computational power, computer vision methods have become popular to solve the localization problem. Subsequent camera images taken during motion are used to determine the transformation between them, and many experiments in various configurations have been made. As an example, [Neumann and You, 1999] detect image features and track theses features over multiple images. While the camera sweeps over a scene, features get out of sight at one image boundary, and new ones appear at the opposite boundary, so that the feature list is dynamically changed. In principle, all 6 DOF pose can be determined up to a scaling factor with a camera. The use of cameras for localization became popular in robotics due to their low price, small size, wide availability, and similarity to the human perception. It is difficult, however, to reliably reconstruct 3D information from 2D images, and the need for calibration and the sensitivity of vision-based approaches to erroneous feature detection are only examples for the issues involved. Algorithms to estimate a 6 DOF out of the 2D camera images are complex and difficult, resulting in large computation time and requiring energy consuming high-end PCs on the battery-powered vehicles. Although cameras are still used in various systems, their importance for localization in mobile robotics has been fading since the mid 1990's due to the introduction of 2D laser scanners.

Today, the vast majority of autonomous vehicles use horizontally mounted 2D laser scanners to detect and avoid obstacles, and to localize themselves. These scanners have the advantage that they provide directly the 2D coordinates of obstacles in the scanner's coordinate system without complex processing, thus reducing computation time and enabling high update rates. They are more reliable for detecting objects, have a large field

of view, and high accuracy. Distance is measured by a laser beam, which is for a horizontally mounted 2D scanner subsequently deflected to multiple directions parallel to the ground by a rotating mirror. Since the angle of the mirror and hence direction of the laser beam is known, it is trivial to compute the 2D location of the point where the laser beam has hit an object; this point is called a scan point. The obtained scan points lie on the outline of the objects, however, naturally only surfaces directly visible from the scanner's location and within its sensing range can result in scan points. All other surfaces do not appear in the scan.

If scans are taken from different positions, they may capture different surfaces or sides of an object; eventually the scans do not even resemble each other, but are rather complementary. However, if the scans are taken from nearby positions, it is likely that a surface visible in one scan is also visible in the other one; in this case, the two scans highly resemble each other. This is typically the case for the horizontal laser scans that we record during our data acquisition. As previously calculated, the translation between subsequent scans is less then 15 cm and the rotation is less than one degree. Since this is small compared to the typical dimensions and distance of buildings in the scans, the perspective remains similar even over multiple scans. Figure 4-2 shows two horizontal laser scans taken at different times t_0 and t_1. The laser rays are drawn as line segments, originating in the scanner's center and ending at the scan point where they hit an obstacle, e.g. a wall. The vehicle has moved between t_0 and t_1, and accordingly, the objects visible in the scan have shifted in the scanner's coordinate system. Even though the two scan in this example are several scanning cycles apart from each other, it can be seen that the formation of the scan points, i.e. the ends of the laser rays, is similar in both scans. Supposed we know the exact pose for one scan, this similarity can be exploited to estimate the pose from which the other scan was taken: the scan from the unknown position is rotated and translated in such a way that maximum congruence with the scan from the known pose emerges. This process is called *scan matching*, and where necessary for clarity, we refer to the particular case where one scan is matched to a second scan more specifically as *scan-to-scan matching*; this is to distinguish it from methods discussed later, where a scan is matched to edges of an aerial photo or a DSM.

In Figure 4-3, the idea of scan matching is illustrated. In Figure 4-3(a), the two scans from Figure 4-2 are drawn in the same coordinate system, with the first one as the reference scan, and the second one as the scan to be matched. After applying an appropriate rotation with angle $\Delta\varphi$ determined in Figure 4-3(b), and an appropriate translation with $(\Delta u,\Delta v)$ determined in Figure 4-3(c), the two scans have maximum congruence, as shown in Figure 4-3(d). Several properties of matched scan pair in Figure 4-3(d) can be noted: First, the two scans are similar, but not exactly congruent: some object surface parts were only visible in either of the scans, but not in the both, due to the slightly different perspective. Second, the locations of the scan points on the surfaces differ, hence correct matching does not imply scan points to be located perfectly on top of each other. Some of the matching algorithms summarized in the following require point-to-point correspondences, and in sparse scans, perfect one-to-one correspondences are not given. Third, each individual scan has an inherent order of points, defined by the

scanning angle, and therefore neighbor relationships are well defined for each scan point. This advantageous property is lost if the point sets are united, thus making operations such as line or feature extraction more complex.

Several approaches have been developed for automatically registering two scans with each other. In [Cox, 1991], Cox suggested to match scan points from a laser scanner with the lines of a manually created a-priori-map of an indoor environment. He obtained an initial estimate for a robot's pose from odometry readings, and used scan-to-line matching to correct for small position errors. Since the initial position estimate is close to the actual position, scan points are close to their corresponding map lines and can be assigned to it. A scan match quality measure is computed as the sum of each scan point's square distance to the corresponding line; the optimum is found with the least square method. Problematic for this method is that few incorrect point-to-line correspondences, for which the distance is large, can adulterate the least square computation and result in an erroneous position estimate. Furthermore, in its proposed form, this approach is dependent on an a-priori map and limited to polygonal environments. Nevertheless, we will come back to this idea for the global localization in Chapter 5.

However, with a small and intuitive modification, this algorithm can also be extended to compute the match between two scans: lines can be extracted from the first scan, and instead of a CAD drawing of the floor, these lines are used as the a-priori reference map for matching a second scan. Gutmann and Schlegel proposed the use of a line filter for both reference and second scan, so that only collinear points are used for the matching process [Gutmann and Schlegel, 1996], based on the assumption that collinear points are likely to be stationary wall points, whereas single isolated points are likely to originate from various other objects, e.g. moving people. They furthermore proposed another heuristic rule to filter out scan point pairs with incorrect correspondences: points for which the distance exceeds a threshold and the most distant twenty percent of the points are not considered. Both modifications to the original approach intend to avoid incorrect correspondences, however, their extension heavily relies on polygonal environments.

Lu and Milios proposed a different matching scheme, the IDC algorithm (iterative dual correspondence, [Lu and Milios, 1994]), similar to the ICP algorithm (iterative closest point, [Besl and McKay, 1992]). Accordingly, their algorithm is able to match two scans directly without the need of features such as lines, but the preliminary is that the relative transformation between the two scans is initially approximately known. In an iteration cycle, a rule independent from the actual rotation and translation provides a guess for the scan point correspondences, and based on these correspondences, the relative transformation is computed and applied. Instead of Besl and McKay's *closest-point* rule, Lu and Milios suggested the *matching-range-point* rule, which additionally considers rotation effects and increases the convergence speed of the algorithm drastically. They also suggested a way to estimate the covariance matrix of the pose estimates, which is necessary for an effective fusion with data from other sensors, e.g. by means of the Kalman filter.

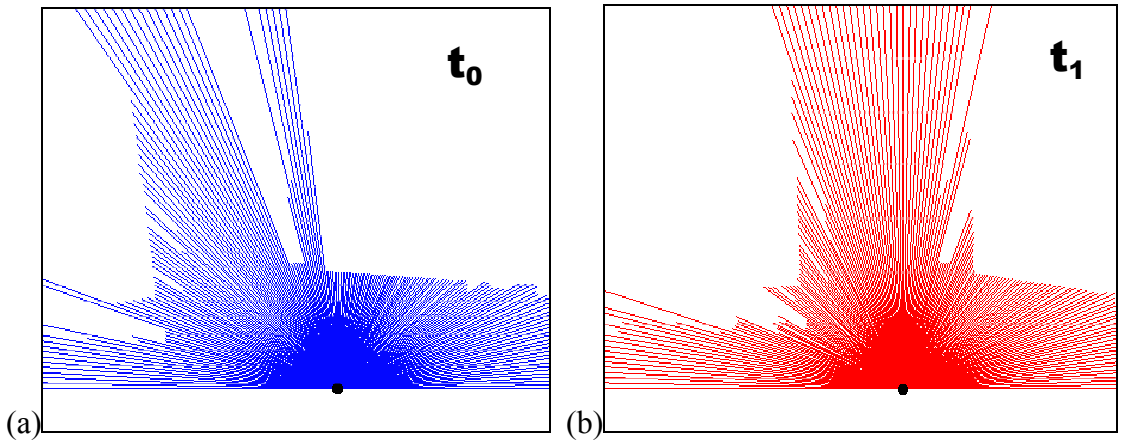


Figure 4-2: Two horizontal laser scans taken at different times t_0 and t_1. The vehicle has moved between t_0 and t_1; accordingly, the objects visible in the scan have shifted in the scanner's coordinate system.

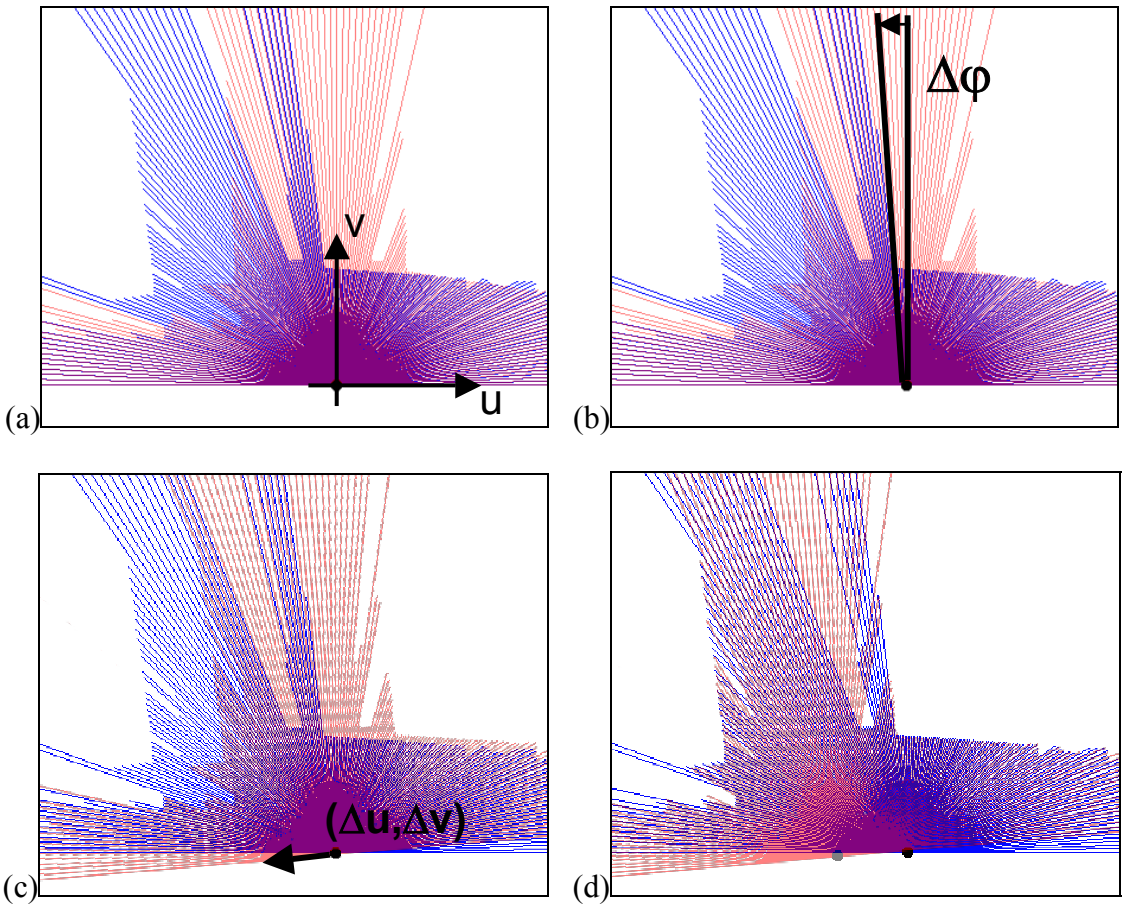


Figure 4-3: Scan matching of two scans taken at different times t_0 and t_1: (a) second scan overlaid on top of the first scan; (b) determining necessary relative rotation; (c) after applying rotation, determing necessary translation; (d) after applying translation, the two scans match and the relative transformation (Δu, Δv, $\Delta\varphi$) between the scans is determined.

A statistical approach was introduced in [Weiss et al., 1994]. Both scans are transformed into a stochastic histogram representation; the basic idea is that the statistical distribution of relative single scan point positions towards each other is preserved during rotation and translation, and therefore the shape of the histograms is invariant. For example, if one creates a histogram with bin size of Δx for the x-coordinates of a set of scan points, and compares it with a histogram of the same points shifted by a translation $n \cdot \Delta x$ along the x axis, it is apparent that the pattern of peaks is absolutely identical, except that the latter histogram is shifted by n bins. Similarly, the shape of a histogram of the orientation angles between subsequent points of a scan is invariant to rotations. If two scans overlap for the most part, this property can be used to determine the transformation parameters between them. The displacement between the histograms of the two scans, and hence the corresponding transformation parameter, is computed by cross correlation. The shape of the angle distribution is invariant to both rotations and translations, whereas the coordinate distribution is not independent from the rotation. Hence, it is necessary that the angle histogram is computed first, and the relative rotation angle is determined and corrected, before the coordinate histograms for the x- and y-direction of each scan are computed and correlated.

This approach works well if the environment is polygonal or contains at least contiguous surfaces. It has the advantage, that the complexity for the histogram generation is only O(n) for n scan points, and the cross correlation is even independent of the number of points, in contrast to the previous approaches, which need an $O(n^2)$ computation time in their original form and are hence less advantageous for real-time applications. However, some of its drawbacks prohibit this method for our application: For noisy, cluttered scans such as resulting from a tree, the orientation angles become rather random, and determining rotation becomes difficult. Even more severe, the accuracy of the parameter estimate is limited to the bin size of the histogram. This bin size cannot be chosen arbitrary small, since then the histogram becomes too sparse. For our application, real-time computation is not necessary, but accuracy is crucial. However, in the next Chapter, we use the idea of an orientation angle histogram for the estimation of the vehicle's rolling angle from vertical scans.

We suggest a scan matching approach related to the one of Cox and Gutmann, but with three major modifications: First, we use both lines and single points of the reference scan for matching, second, we do not treat the problem of erroneous correspondences separately, rather we consider it directly in the computation of a match quality function by using robust least squares, and third, we compensate for effects of finite scanning time on the scan points, which is a problem specific to our high-speed acquisition.

Since scan points of the reference scan do not necessarily correspond directly to points in the second scan, but often lie in intermediate positions, it is advantageous to match points not directly to points, but rather to extract lines in one scan, e.g. the reference scan, and match the points of the second scan to the lines. However, in contrast to many indoor navigation situations, where there is always a sufficient number of lines visible in each scan, the scenery in urban environments is more complex because of many non-planar

objects such as trees, masts, cables, and partially reflecting windows. These objects provide additional if not at times the only information about the relative pose of successive scans, and as such it is desirable to use them for pose estimation.

Hence, we apply the following algorithm for obtaining line segment approximations to the reference scan: We connect successive scan points to form a line strip, if the difference between their depth values does not exceed a depth dependent threshold. More specifically, our algorithm can be summarized as follows in pseudo-code:

Create set of lines:

```
for angle: = 0 to 180
  {
    point1: = (cos(angle), sin(angle))*distance(angle);

    if  abs( distance(angle) - distance(angle+1)) < maximum_discontinuity)
      {
        point2: = (cos(angle+1), sin(angle+1))*distance(angle+1);

        AddLineSegment from point1 to point2;
      }
    else
      AddLineSegment from point1 to point1;
  }
```

Accordingly, a single isolated scan point is approximated by a "degenerated" line segment, i.e. a point and considered during the subsequent computations. While a disadvantage of this method is that straight lines are not computed by a least squares fit over multiple points, the advantage is that curved objects, such as trees, and small objects, such as masts, cables, are also used for matching. Figure 4-4 shows rays of a laser scan in gray and the line segment approximation in black, and as seen, more features than only long straight lines are extracted.

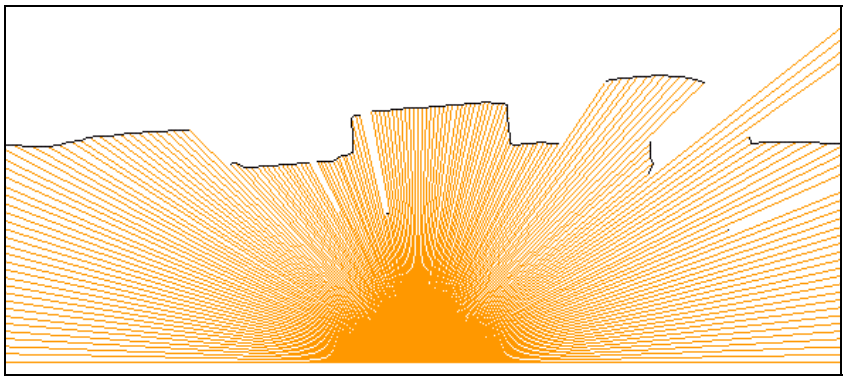


Figure 4-4: Scan and its line segment approximation

The line strip approximation of the reference scan is used as a map to register the second scan. Therefore, we introduce a local coordinate system [u,v] implied by the reference scan, with the sensor module at its center. The u-axis is aligned with the truck's principal axis, while the v-axis is orthogonal to it, with the positive v-axis pointing towards the left side of the truck, as shown in Figure 5. The scanner provides angle and distance for each scan point, enabling us to compute its Cartesian coordinates by using simple trigonometric functions.

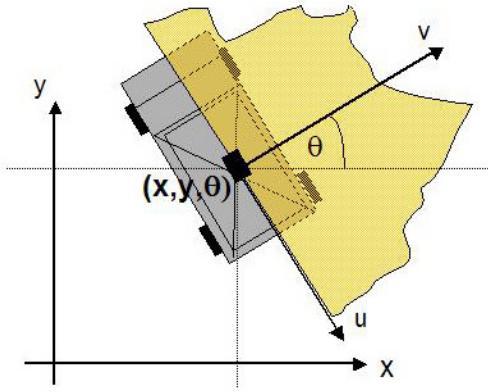


Figure 4-5: Local and global coordinate system

To match two scans, we maximize a function that computes the quality of alignment $Q = f(\Delta u, \Delta v, \Delta\varphi)$ for a given displacement Δu, Δv and a rotation $\Delta\varphi$ of the scans against each other. Therefore, we perform the following steps: First we compute a set of lines l_i from the reference scan points as described before. Given a translation vector $\vec{t} = (\Delta u, \Delta v)$ and a 2x2 rotation matrix $R(\Delta\varphi)$ with rotation angle $\Delta\varphi$, we transform the points p_j of the second scan to the points p'_j according to

$$\vec{p}'_j(\Delta u, \Delta v, \Delta\varphi) = R(\Delta\varphi) \cdot \vec{p}_j + \vec{t} \; . \qquad \textbf{(4-1)}$$

Then we compute for each point $\vec{p'}_j$ the Euclidean distance $d(\vec{p'}_j, l_i)$ to each line segment l_i and set d_{min} to:

$$d_{\min}(\overrightarrow{p'}_j(\Delta u, \Delta v, \Delta\varphi)) = \min_i \left\{ d(\vec{p'}_j, l_i) \right\}. \qquad \textbf{(4-2)}$$

Intuitively, d_{min} is the distance between p'_j and the closest point on any of the lines in the reference scan. Distance measurement errors of the scanners can be modeled as Gaussian; however, in order to suppress erroneous point-to-line correspondences, we do not use the simple sum of distance squares, rather, we use a formula known as robust least squares [Triggs et al., 2000] and compute Q as follows:

$$Q(\Delta u, \Delta v, \Delta\varphi) = \sum_j \exp\left(-\frac{d_{\min}(\vec{p'}_j(\Delta u, \Delta v, \Delta\varphi))^2}{2 \cdot \sigma_s^2} \right) \qquad \textbf{(4-3)}$$

where σ_s^2 is the variance of the laser distance measurement, specified by the manufacturer. This formula takes into account the distribution of the distance measurement values, while suppressing deviations beyond this distribution as outliers. It has least squares-like behavior in the near range but does not take into account points that are far away from any line. The block diagram of this quality computation is shown in Figure 4-6.

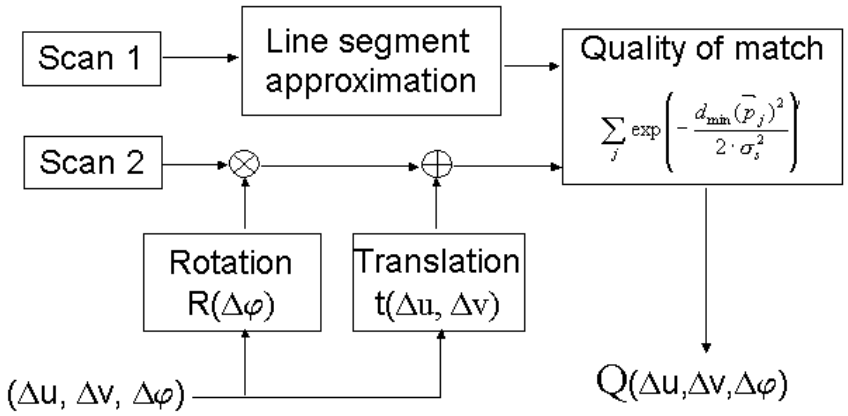


Figure 4-6: Block diagram of quality computation

The parameters (Δu, Δv, Δφ) for the best match between a scan pair are found by optimizing Q. Steepest decent search methods have the advantage of finding the minimum fast, but due to noise and erroneous point-to-line assignments, they can become trapped in local minima if not started from a "good" initial point. Therefore, we use a combined method of sampling the parameter space and discrete steepest decent, where we first sample the parameter space in coarse steps and then refine the search around the minimum with steepest decent. As outlined in the following pseudo code, this function delivers an estimate (best_d.u, best_d.v, best_d.φ) for the necessary transformation to align the two scans best:

Find optimal scan match:

```
/* parameter vectors {u,v,φ} for our setup */
searchrange:= {150 cm, 20 cm, 10 degree};
sampledensity:= {10 cm, 10 cm, 2 degree };
stepsize: = {0.2 cm, 0.2 cm, 0.01 degree};

/* First step: Sampling parameter space */
best_d := {0,0,0};

for d.u = -searchrange.u  to  searchrange.u  step  sampledensity.u
  for d.v = -searchrange.v  to  searchrange.v  step  sampledensity.v
    for d.φ = -searchrange.φ  to  searchrange.φ  step  sampledensity.φ
        if Q(d) < Q(best_d) then
           best_d := d;


/* Second step: steepest decent */

direction := {0,0,0};

do {
    if Q(best_d) < Q(best_d + {stepsize.u,0,0})  then  direction.u := stepsize.u;
    if Q(best_d) < Q(best_d – {stepsize.u,0,0})  then  direction.u := –stepsize.u;

    if Q(best_d) < Q(best_d + {0,stepsize.v,0})  then  direction.v := stepsize.v;
    if Q(best_d) < Q(best_d – {0,stepsize.v,0})  then  direction.v := –stepsize.v;

    if Q(best_d) < Q(best_d + {0,0,stepsize.φ})  then  direction.φ := stepsize.φ;
    if Q(best_d) < Q(best_d – {0,0,stepsize.φ})  then  direction.φ := –stepsize.φ;

    best_d := best_d + direction;

} while direction != {0,0,0};

return best_d;
```

Acquisition time for one single scan is quite small, and since the previous approaches were developed for slow moving robots, none of them has considered motion *during* a scan. However, our sensor unit is mounted on a fast moving truck, and hence intra-scan motion can reach noticeable levels, and we can increase accuracy by compensating for it. The acquisition time for one 180-degree-scan is 6.667 ms; for a maximum angular speed of 45 degree/sec and a maximum velocity of 25 mph, this results in a maximum rotation of 0.3 degrees and a maximum displacement of 7.4 cm between the first scan point at 0 degree and the last scan point at 180 degree. Dependent on the scanner's rotation

direction in regard to the motion, the entire scan is linearly stretched or stitched along an axis in case of a translation, and along the fan in case of a rotation, respectively. The effect can come into the range of the accuracy we intend to achieve, and while not crucial for one single scan, these errors can accumulate to a substantial amount for a long drive.

In the stationary case, the Cartesian coordinates $(u,v)_i$ of the i^{th} point of the scan is

$$\begin{pmatrix} u \\ v \end{pmatrix}_i = d_i \cdot \begin{pmatrix} \cos(\Delta\psi \cdot i) \\ \sin(\Delta\psi \cdot i) \end{pmatrix} \qquad \textbf{(4-4)}$$

where d_i is the measured distance and $\Delta\psi$ is the angle increment between adjacent measurements in a scan. Supposed we know the current angular speed Ω_T and the speed $\vec{V}_T$ of the truck, and assume it as constant during the short time the scan is taken, we introduce a correction term for the dynamic case, so that the Cartesian coordinates are computed according to

$$\begin{pmatrix} u \\ v \end{pmatrix}_i = d_i \cdot \begin{pmatrix} \cos((\Delta\psi - \Omega_T \cdot \Delta t) \cdot i) \\ \sin((\Delta\psi - \Omega_T \cdot \Delta t) \cdot i) \end{pmatrix} + \vec{V}_T \cdot \Delta t \cdot i \qquad \textbf{(4-5)}$$

where $\Delta\psi$ is again the angle increment and Δt the time elapsed between adjacent measurements in a scan. For our scanner, the values are $\Delta\psi$ = 1 degree and Δt = (1/75Hz)/360 = 37 μs.

The question is now how to obtain an estimate for Ω_T and $\vec{V}_T$. Instead of reading these variables from additional external sensors, we compute the estimates conveniently as a byproduct during the scan matching process: The time ΔT elapsed between the two scans of a pair to match is given by the difference of their ordering number. Then, the speed $\vec{V}_T$ can simply be computed by dividing the translation by ΔT, and the rotational speed Ω_T by dividing the rotation by ΔT, respectively. One possibility is therefore to iteratively first compute the optimal scan match without correction, then use the computed transformation to correct the scan points, and then compute the scan match again and so on. Since the correction is small compared to the transformation, it is sufficient to iterate only once. However, even one single iteration would double the computation time, and it is desirable to avoid this increase. In the next Section, we will reconstruct the driven path by matching series of subsequent scans; since the car's motion does not change substantially between the single scan pairs, it is feasible to use the Ω_T and $\vec{V}_T$ estimates from the previous pair as a valid estimate for the current scan match.

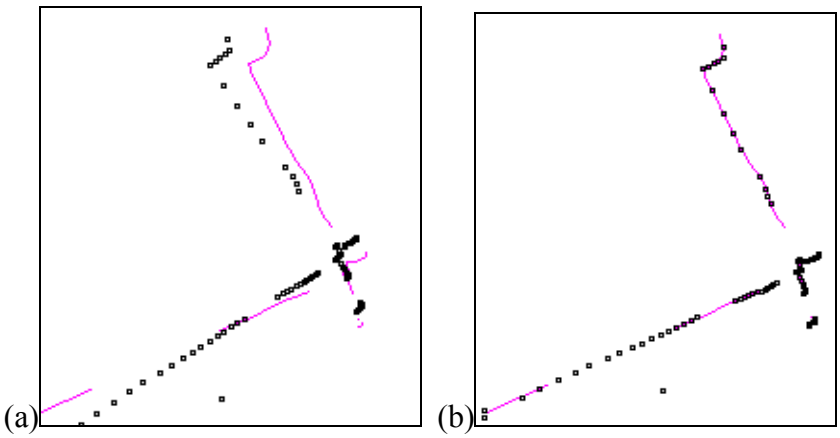


Figure 4-7: Matching scan points and line segment approximation; (a) before and (b) after match.

Figure 4-7 shows the result of a scan match performed with the described algorithm, by comparing a scan pair before and after matching. The line segment approximation of the reference scan is drawn in gray, and the points of the second scan are drawn in black, respectively. While the shown example is a scan in a downtown area containing some smooth surfaces, it is obvious that the accuracy of the matching process can depend significantly on the particular environment. In areas with buildings and therefore many straight lines in the scans, the matching is likely to work best; in areas where the scanner hits mainly the leaves of trees, scan points are more random. Hence, the quality of the scan match is lower and the maximum is eventually not very sharp. We have found that in this case the accuracy may be lower, but due to utilization of non-collinear features still in an acceptable range for determining the motion reliably, in contrast to most of the existing algorithms previously summarized. Note that in the worst case, if there is an entirely featureless view such as the one on an empty parking lot or on a freeway ramp, it is not possible to estimate the relative motion at all. Fortunately however, usual downtown environments are dense and full of features, so that using scan matching to track pose is quite reliable. In particular, as it is our goal to reconstruct building structures, pose estimation by scan matching is almost an ideal solution, since it provides the most accuracy for the areas that are the most interesting for us. Further discussion about the accuracy of the matching process and the pose reconstruction follows in the next section.

4.2 Path Computation

The scan matching algorithm introduced in the previous section determines directly the translation (Δu, Δv) and the rotation $\Delta\varphi$ between two given scans. In the following, we refer to the relative pose between a scan pair as a relative *step*; thus each step S_i consists of a Δu_i, Δv_i, and $\Delta\varphi_i$ estimate in the local coordinate system. We can obtain an initial estimate for the path, i.e. the trajectory traversed during data acquisition, by defining a starting pose and successively concatenating relative steps between subsequent scans.

Since the scan matching provides only a 2D estimate, the resulting path is in a plane; this represents pose entirely if our environment is flat without significant hills.

In this case, we can describe the global 2D pose of the sensor module by the parameter tuple (x, y, θ), where x and y are the Cartesian world coordinates, and θ is the yaw angle, i.e. the orientation of the truck, as shown in Figure 4-5. If speed V(t) and orientation θ(t) of the truck is known, the motion of the rear axis of the truck can be described as:

$$\begin{aligned} x(t+\Delta t) &= x(t) + V(t)\cdot\Delta t\cdot\cos(\theta(t)) \\ y(t+\Delta t) &= y(t) + V(t)\cdot\Delta t\cdot\sin(\theta(t)) \end{aligned} \qquad \textbf{(4-6)}$$

The sensor module is not mounted above the rear axis, but in the middle of the truck, so that during motion along a curve, it experiences a motion component not only along, but also orthogonal to the truck's principal axis. As an estimate for the relative position change (Δu_i, Δv_i, $\Delta\varphi$) of the sensor module in its local coordinate system [u,v] is obtained for each step S_i, we can compute the global positions (x_i, y_i, θ_i). To compute the 2D path, we start with an initial position (x_0, y_0, θ_0), perform for each step S_i a scan match to obtain (Δu_i, Δv_i, $\Delta\varphi$), and apply the coordinate transformation, so that the new position (x_{i+1}, y_{i+1},θ_{i+1}) can be computed in world coordinates as:

$$\begin{aligned} x_{i+1} &= x_i + \Delta u_i\cdot\cos(\theta_i+\Delta\varphi_i) - \Delta v_i\cdot\sin(\theta_i+\Delta\varphi_i) \\ y_{i+1} &= y_i + \Delta u_i\cdot\sin(\theta_i+\Delta\varphi_i) + \Delta v_i\cdot\cos(\theta_i+\Delta\varphi_i) \\ \theta_{i+1} &= \theta_i + \Delta\varphi_i \end{aligned} \qquad \textbf{(4-7)}$$

For non-level areas, the incline results in an apparent source of error in length: The scan-to-scan matching estimates for the 3-DOF relative motion, i.e. 2D translation and rotation, are given in the scanner's local scanning plane. If the vehicle is on a slope, this local coordinate system is tilted at an angle towards the global (x,y) plane, and hence the translation should strictly speaking be corrected with the cosine of the pitch angle. Fortunately this stretching effect is small; the relative length error is

$$\frac{\Delta l_{err}}{l} = 1-\cos(pitch) \approx 1-(1-pitch^2) = pitch^2, \qquad (4\text{-}8)$$

and while for example 0.5 % for a substantial 10% slope, it is only 0.06% for a moderate 2% slope. Thus, it turns out that this error is easily within the correction capabilities of our global localization to be introduced in the next Chapter. While this effect may decrease accuracy in extremely steep area such as Russian Hill in San Francisco, we can usually safely neglect it. Hence, we utilize the relative estimates from the scan-to-scan matching in the following as if they were given in the global coordinate system. In any case the resulting 2D path is a good *initial* approximation even for hillside areas, since the principal motion of a vehicle is horizontal. In the next Chapter, we will describe an

extension that uses an airborne digital surface model and complements the 2D pose to a full 3D pose.

Since errors in the estimation accumulate with each iteration step of equation 4-7, it is important to recover the path with as few steps as possible by subsampling the scans by a large factor; this is especially necessary when the truck is stationary due to traffic conditions. On the other hand, it is desirable to use scans that are taken from nearby positions, so that perspective change is small and overlap between scans is sufficient for accurate matching; this requires the subsampling factor to be small. Therefore, we need to find a compromise between these conflicting requirements.

The scanner takes horizontal scans at a frequency of 75 Hz, and with the maximum city driving speed of 25 miles per hour, the maximum relative displacement for successive scans is

$$\Delta u_{max} = 25 \text{ mph}/75 \text{ Hz} = 14.8 \text{ cm}. \quad \textbf{(4-9)}$$

For typical distances in our measurement scenario, we have found that perspective and insufficient scan overlap can become critical issues for position changes of more than 2 meters. To strike a balance between large and small subsampling factors, one could define a fixed standard subsampling factor, which is determined based on the assumed maximum driving speed, e.g. it could be 10 if a maximum displacement of 150 centimeter is allowed, since 10 x 14.8 cm is less than 150 cm. While this defines the upper bound of the displacement, it does not constitute a lower bound, e.g. the displacement is zero if the vehicle stands temporarily still. This can be solved by using not a fixed subsampling factor, but rather to adapt it to the driving speed, so that estimated displacements between successive matched scans is e.g. between 80 and 150 cm. This not only increases the accuracy of the path, but also improves the computational efficiency. We define a minimum displacement for a step, e.g. 80 cm. To compute a new step, we first match the scans that are the standard subsampling number apart from each other; if the resulting displacement is greater than the minimum displacement, we accept the match as the next step, else, we increase the subsampling factor for this pair. We choose the next scan candidate based on extrapolating the scan number according to the ratio of current displacement and minimum displacement, and perform the scan matching with this new pair. The result is a path with the size of all steps in the desired range.

Since we do not have the ground truth, it is difficult to determine the actual correctness for the driven path and the accuracy of the relative step estimates. However, we can make experiments that give us at least hints in which range the error can be expected: we can use the alignment of the horizontal scan points as a measure for the quality of the scan matching result. Assuming that the surface of the wall is ideally smooth, all scan points of a building wall should lay perfectly on a line, even if scanned from different poses. The two major effect that cause the scan points to be off the ideal line are first, the inherent measurement noise of the distance measurement, and second, the imperfection of the pose computation, i.e. the scan matching. We model the pose estimates

$(\Delta u, \Delta v, \Delta\varphi)_i$ computed in the scan matching process as the sum of the true pose values $\left(\overline{\Delta u}, \overline{\Delta v}, \overline{\Delta\varphi}\right)_i$ and an additional (unknown) error $(e_{\Delta u}, e_{\Delta v}, e_{\Delta\varphi})_i$ with a Gaussian distribution, so that we can write

$$\left(\Delta u, \Delta v, \Delta\varphi\right)_i = \left(\overline{\Delta u}, \overline{\Delta v}, \overline{\Delta\varphi}\right)_i + \left(e_{\Delta u}, e_{\Delta v}, e_{\Delta\varphi}\right)_i \qquad \textbf{(4-10)}$$

If we add to each of the parameters Δu_i, Δv_i, and $\Delta\varphi_i$ artificial Gaussian noise $(n_{\Delta u}, n_{\Delta v}, n_{\Delta\varphi})$ with a standard deviation $(\sigma_{n,\Delta u}, \sigma_{n,\Delta v}, \sigma_{n,\Delta\varphi})$, we obtain distorted estimates

$$\left(\Delta u^*, \Delta v^*, \Delta\varphi^*\right)_i = \left(\Delta u, \Delta v, \Delta\varphi_i\right)_i + \left(n_{\Delta u}, n_{\Delta v}, n_{\Delta\varphi}\right)_i . \qquad \textbf{(4-11)}$$

Then, we re-compute path and scan points with the distorted estimates, and inspect visually the effect on the alignment. If the added noise $(n_{\Delta u}, n_{\Delta v}, n_{\Delta\varphi})$ is well below the inherent estimation error $(e_{\Delta u}, e_{\Delta v}, e_{\Delta\varphi})$, there is no apparent effect on the path; however, if the distortion comes into the same order of magnitude or exceeds the matching noise, the alignment of the horizontal scan points degrades noticeably. Therefore, our method to determine the inherent error is to add different levels of noise and to observe from which level on the lines start blurring. At this level, the added distortion and the inherent errors are approximately equal. Figure 4-8 shows a piece of the computed path, which the vehicle traversed while scanning the half plane to the right, indicated as a series of steps (gray arrows). From each position at the end of an arrow, a horizontal scan has been taken, and the horizontal scan points corresponding to these positions have been superimposed (black). The shown area is a typical city environment where both buildings and trees are present, visible as collinear points and point clusters, respectively. In order to investigate the effect of adding the Gaussian distortion on the alignment, we closer examine the points in the marked square in the next figures.

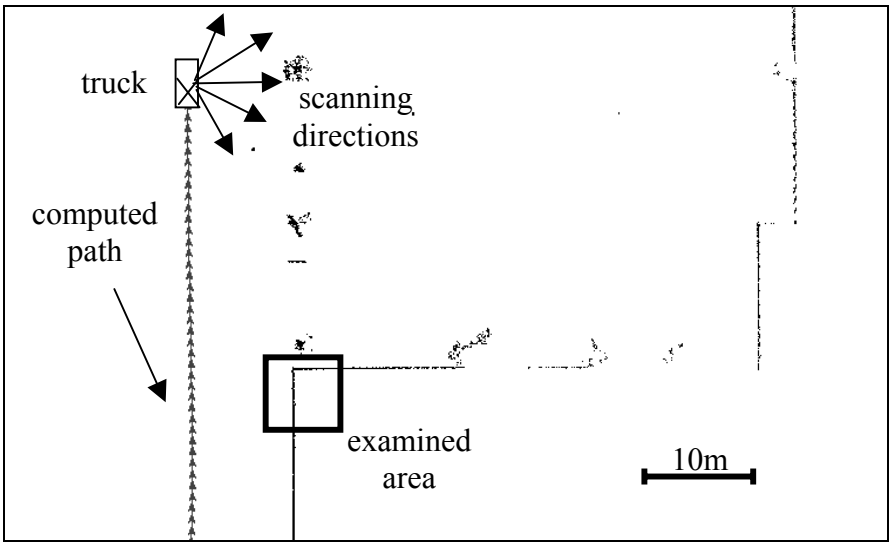
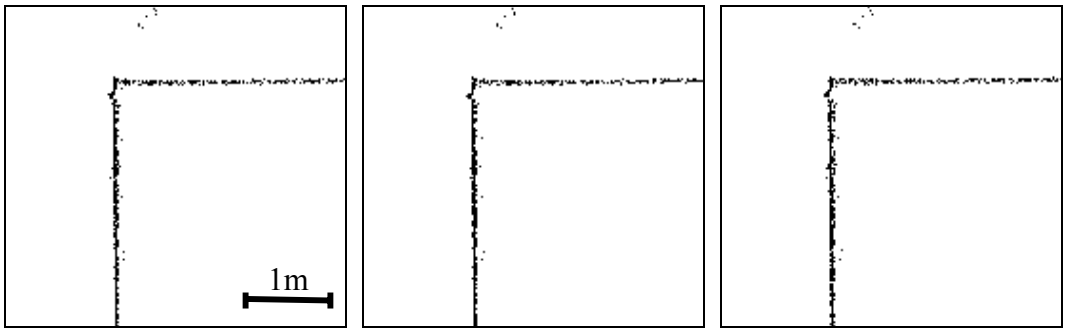
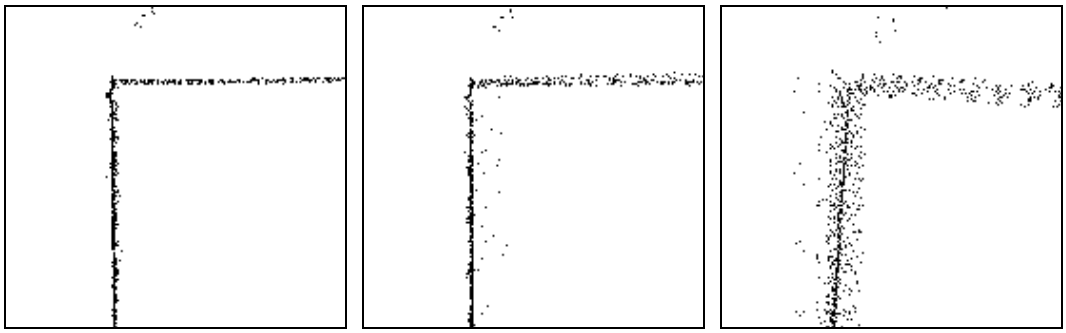


Figure 4-8: Computed path and overlaid horizontal scan points; the steps of the path are indicated as arrows (gray). From the each position at the arrow tip a horizontal scan has been taken, and all scans are drawn from this position. The square zoom clip is examined in the next figures.

Figure 4-9 shows the effect of adding distortion to the $\Delta\varphi_i$ estimates, Figure 4-10 to the Δu_i estimates, and Figure 4-11 to the Δv_i estimates, respectively; in each case the distortion is only added to the specified parameter and not to the others. As seen in Figure 4-9, the point alignment starts to degrade visibly for an angle distortion between $\sigma_{\Delta\varphi} = 0.01^\circ$ and $\sigma_{\Delta\varphi} = 0.03^\circ$. For the translation parameters Δu and Δv, this degradation starts between $\sigma = 0.3$ cm and $\sigma = 1.0$ cm. Note that for the translation parameters, the alignment only degrades along the direction the distortion is added. For the shown example, the vehicle trajectory is approximately vertical in the figure; since the scanner's u-axis points along the driving direction and the v-axis is perpendicular to it, the degradation is approximately vertical for Δu_i distortion and horizontal for Δv_i distortion.

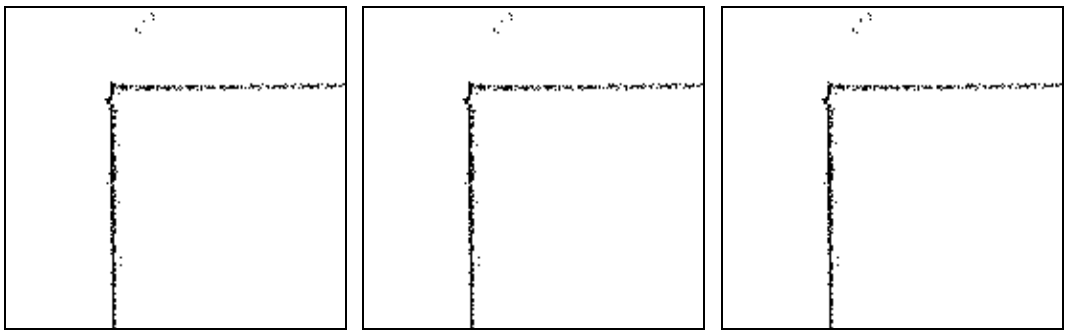


(a) no noise added **(b) $\sigma_{\Delta\varphi} = 0.003^\circ$** **(c) $\sigma_{\Delta\varphi} = 0.01^\circ$**

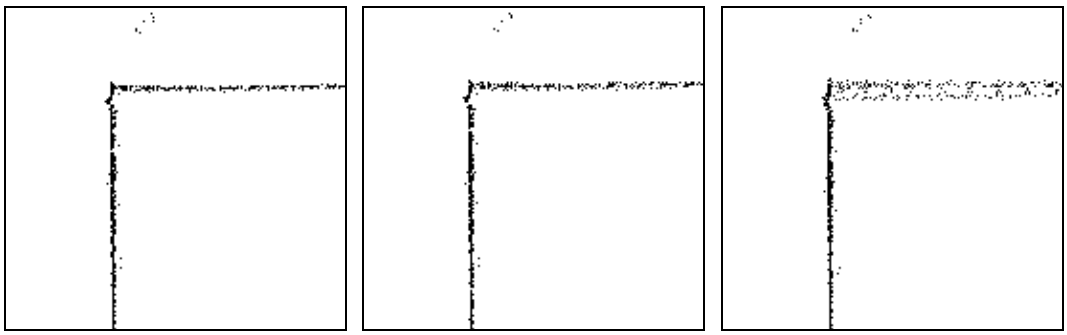


(d) $\sigma_{\Delta\varphi} = 0.03^\circ$ **(e) $\sigma_{\Delta\varphi} = 0.1^\circ$** **(f) $\sigma_{\Delta\varphi} = 0.3^\circ$**

Figure 4-9: Alignment of scan points after adding different levels of Gaussian white noise with a standard deviation $\sigma_{\Delta\varphi}$ to the rotation angle estimates $\Delta\varphi_i$; the point alignment decreases visibly for a distortion with $\sigma_{\Delta\varphi} > 0.01^\circ$.

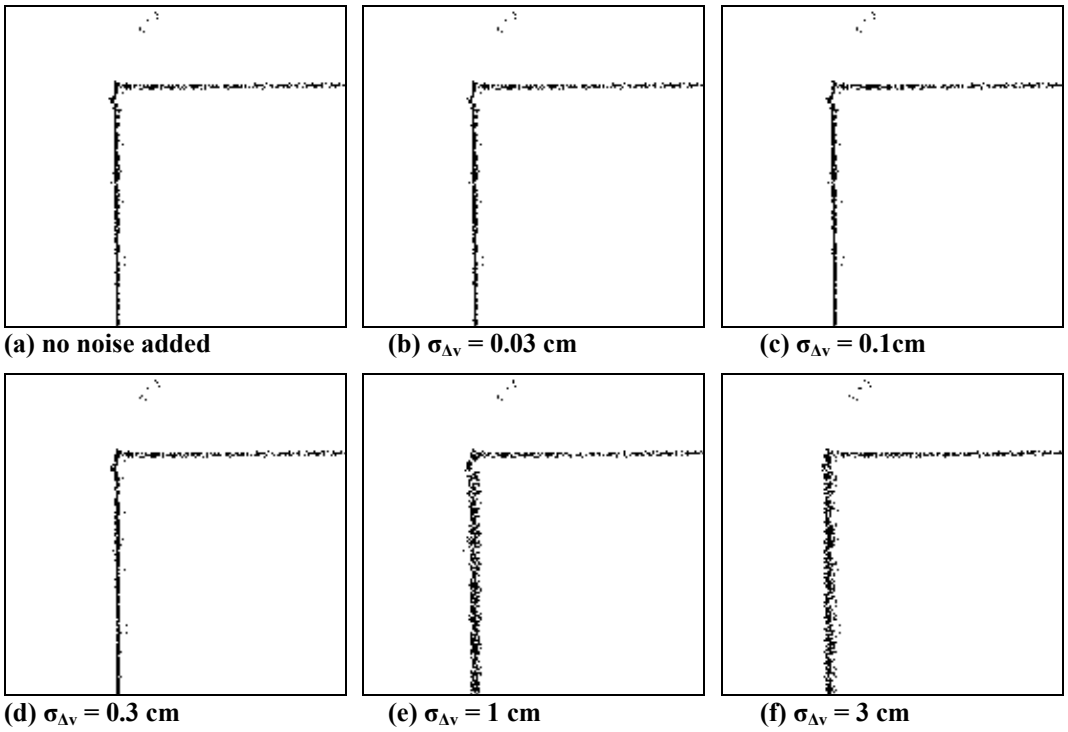


(a) no noise added **(b) $\sigma_{\Delta u} = 0.03$ cm** **(c) $\sigma_{\Delta u} = 0.1$cm**



(d) $\sigma_{\Delta u} = 0.3$ cm **(e) $\sigma_{\Delta u} = 1$ cm** **(f) $\sigma_{\Delta u} = 3$ cm**

Figure 4-10: Alignment of scan points after adding different levels of Gaussian white noise with a standard deviation $\sigma_{\Delta u}$ to the Δu_i estimates; the point alignment in vertical direction decreases visibly for a distortion with $\sigma_{\Delta u} > 0.3$ cm.



(a) no noise added **(b) $\sigma_{\Delta v}$ = 0.03 cm** **(c) $\sigma_{\Delta v}$ = 0.1cm**

(d) $\sigma_{\Delta v}$ = 0.3 cm **(e) $\sigma_{\Delta v}$ = 1 cm** **(f) $\sigma_{\Delta v}$ = 3 cm**

Figure 4-11: Alignment of scan points after adding different levels of Gaussian white noise with a standard deviation $\sigma_{\Delta v}$ to the Δv_i estimates; the point alignment in vertical direction decreases visibly for a distortion with $\sigma_{\Delta v} > 0.3$ cm.

From these experiments, we conclude that for typical city areas, which contain a sufficient amount of buildings, the accuracy of the scan matching process is better than 0.03 degrees for the rotation and better than 1 cm for the translation parameters. In other words, within a local environment of a few relative steps, e.g. the length of a building, the estimated pose is extremely accurate. As a result, points of subsequent scan columns can be registered correctly in respect to each other, and this local pose accuracy allows the reconstruction of an accurate 3D model for a building. However, it can be noted that there are remaining finite errors in the estimates, and mismatches may decrease accuracy further in areas such as empty parking lots or public parks, which do not contain appropriate surfaces. We will discuss the effect of these errors for longer drives and suggest a possible solution in the following section.

5 Global Localization

Even small errors in the single relative estimates accumulate to significant global position errors, when many subsequent relative steps are added over a longer period. Especially inaccuracies in the relative angle estimates contribute to increasingly catastrophic global position errors. From a signal theory viewpoint, it appears that the "high frequency" components of the reconstructed initial path, e.g. overtaking maneuvers or turns, are correctly recovered, but there are errors in the "low frequency" components, so that resulting absolute position has a drift-like behavior. In Figure 5-1, a downtown Berkeley path reconstructed from the relative position estimates as described in the previous section is shown superimposed on a digital roadmap. This path starts at a known position, but follows the road only for the first few hundred meters acceptably; while driving continues, it is more and more off. In particular, there are relatively large error at some locations, e.g. in the first turn where the scanner has faced Sproul Plaza, an area with many trees and almost no buildings in the field of view of the scanner. As seen, this error is propagated through the entire path segment following the turn.

Figure 5-1: Path obtained by adding relative steps overlaid on top of a digital road map. During the first few hundred meters, the path follows the road map; the longer the driving, the more the path separates from the map, and it is finally completely off the real road.

This example illustrates the necessity of correcting pose globally, and we will describe two approaches to obtaining an estimate for the absolute pose in the following. Since we want to keep the high local precision of the scan matching, which enables us to assemble subsequent scans consistently, we combine the advantages of both, i.e. we use the scan-to-scan matching locally, while correcting the position with absolute estimates globally.

5.1 Background and Related Work

One of the oldest localization approaches is the navigation based on landmarks, and where natural landmarks were not available or visible, artificial ones such as beacons have been placed at known positions. During the centuries, beacons have changed from fires in ancient marine navigation to radio signal transmitting stations. For indoor applications, various types have been utilized, including sonic or laser beacons, corner cubes, reflectors, bar-codes, spot marks, and many others. A sensor on the vehicle detects the angle and eventually the distance at which a beacon signals is received. Sensors can be ultrasonic receivers for ultra sound beacons, photo diodes for laser beacons or reflectors, and vision recognition systems. Then, the position is determined using simple geometry. For example, [Leonard and Durrant-Whyte, 1991] extract beacons consisting of planes, cylinders, and corners from sonar scans and match them with landmarks in a geometric map of the environment. However, this approach is not suitable for an urban environment. First, buildings block the direct view to beacons, so that they are not visible in most locations, and second, setting up an environment is a tremendous effort in both timely and financial regard; practically, it is not realizable to install enough beacons to completely cover the area of an entire city.

For navigation in outdoor environments, the Global Positioning System (GPS) has become the standard source for position estimation. Originally developed by the U.S. Department of Defense as a military system, GPS has become a global utility since the mid 1980s. It provides global 3D position in form of latitude, longitude, and altitude, and benefits users around the world in many different applications, including air, road, and marine navigation. Twenty-four satellites are positioned around the globe, and each satellite transmits an individual Pseudo-Random-Noise signal. From this signal, a user can determine the distance to each satellite via time-of-flight and find his position by triangulation. In this sense, one can regard GPS as an extremely sophisticated extraterrestrial beacon system. The number of satellites visible from an earth location depends on both location and time of day; at least four satellites are necessary to compute the position. The possible accuracy depends on number and constellation of the satellites, and atmospheric distortion effects. Until May 2000, the Department of Defense also added an intentional distortion called Selective Availability to the signal, in order to limit the absolute position accuracy for civil users to about 100 meters. Accuracy can be significantly increased with differential GPS, or DGPS. In this setup, a second GPS receiver is placed at a know position, for example on a USGS (United States Geological Survey) landmark. The various sources of error result in a total time-dependent offset between the known reference position and its GPS estimate; this offset is then used to correct the GPS estimates for the unknown position of the user.

However, even DGPS cannot provide continuously reliable localization at centimeter-accuracy in urban environments. In presence of high building and in “urban canyons”, the direct line-of-sight to most satellites is blocked. Due to this occlusion, there are at many locations not enough satellites to determine the position, and therefore no position estimates. Furthermore, there can be multi-path reflections of the satellite signals, e.g.

from building walls, so that the calculated time-of-flight to the satellite and hence the computed global pose is erroneous. Commercial GPS car navigation systems overcome these problems by using motion estimates from odometry, i.e. counting the rotations of the wheels and determining the steering angle. While a typical low-cost car GPS receiver can have a standard deviation of as high as 100 meters and many signal dropouts, the odometry provides continuously relative position estimates at higher precision, but with drift-like behavior. Where available, the absolute GPS estimates are therefore used to correct the relative motion estimates such that the mean difference between position and GPS is zero. This technique can achieve accuracy in the low meter range, which is sufficient for tracking a vehicle in a graph-like digital roadmap. In our localization approach, we will exploit the same idea, i.e. utilizing accurate, but drift-prone local estimates, and correcting them with coarser, but globally correct absolute position estimates.

Some interesting techniques come from the field of autonomous mobile robotics. Since position determination is one of the core problems in this field, multiple map-based localization approaches have been developed, especially for indoor navigation. A map can either be *metric*, i.e. it contains the geometry of objects in exact coordinates and dimensions, or *topological*, i.e. it contains the connectivity of objects or places in a graph at a more abstract level. As an example, the answer for the question "Where is room 307?" could be "10 meter west and 20 meter north from here" if a metric map is used, or "It's the third room on the left side" if a topological map is used. If not explicitly noted otherwise, in this thesis the term "map" will implicitly mean metric map, in accordance to the common usage of this term in geography. Popular representations for metric maps in robotics are *occupancy grids*, where the 2D map space is divided into fine-grained grid cells, and each cell is either marked as occupied or empty, and *polygonal maps*, where outlines of objects are stored as polygons. Early examples of the first type of representation are given in [Elfes, 1987] and [Moravec, 1988], and for the latter one in [Chatila and Laumond, 1985].

Most work has focused on localizing a robot in respect to a preexisting, a-priori map of the environment, in which characteristic features are noted. Features can be explicit landmarks, often entered manually into the map, e.g. lines from existing CAD drawings of a building. Features can also be rather implicit, for example if previous raw measurements are marked in a map without further processing, and e.g. denote the location of a wall. The robot uses its perception, i.e. sensor readouts, to detect and identify surrounding landmarks, and the most popular sensors are cameras, ultrasonic sensors, and laser scanners. The process of explicitly computing and identifying a landmark in the sensor data is not always necessary; some approaches can match raw data directly with the map. This is advantageous if feature extraction is difficult or not reliable. However, even these approaches rely implicitly on features contained in both model and perception. More recent work has addressed the problem of exploring an unknown environment in which both, localization and map building has to be done simultaneously. The transitions between these two scenarios are often fuzzy, for example

if a robot starts with an incomplete map and updates it by adding missing objects or detecting dynamic changes.

There are various strategies to process the perceived input and to determine and represent a robot's pose. Computational expensive feature detection usually requires a search space reduction, which typically uses an approximate pose estimate from a tracking process, i.e. the information about previous positions and cues from other sensors such as odometry readings. Hence, pose and perception mutually affect each other, and it is important how pose and its uncertainty are represented. While in the early days of mobile robotics system state was described by a single parameter set, a representation of its uncertainty has been introduced with the Kalman filter [Kalman, 1960]. In recent years, an entire field called probabilistic robotics has emerged, in which the believed robot pose can be represented by arbitrary probability distribution; we will give a more detailed overview over this field in Section 5.5.

Depending on the believed pose range, the location for which map landmarks and perceived features match the best is determined in a data fusion process. [Schiele and Crowley, 1994] compare several approaches using occupancy grid maps. Ultrasonic sensors are used to localize the vehicle based on the grid map, and the grid is updated if measurements coherently indicate a status change for a cell. As already mentioned, [Cox, 1991] suggested to match scan points from a laser scanner with the lines of a manually created a-priori floor plan. In this case, line segments are the features of a map, and the raw scan points are used to perform a point-to-line match as described in the previous section. [Weiss et al., 1994] go even further in a localization approach also based on laser scans: Neither map nor perception depends on explicit landmarks; rather, scans from several known positions and orientations are taken, or simulated using a line map, and their histogram statistics are stored along with the location in a scan database. Then, during the robot's motion, the histogram of an incoming scan is computed and compared to the histograms in the database, essentially performing a scan-to-scan match as described in Chapter 4. Then, the relative pose to the scan with the most similar histogram in the database is computed, and the absolute pose can be determined with the known pose of the database scan.

If a map of the environment is not available, both map building and navigation must be performed iteratively. The preliminary map is used for estimation of the current position, e.g. by using the techniques described above, and subsequently updated by registration of new features. This straightforward, iterative map building process has pitfalls. Since there are errors in the vector estimates of the relative movement as well as in the map-based localization, new features may be registered at inaccurate or incorrect locations. This yields an inaccurate map, and hence an even more inaccurate pose estimation in the next localization phase, resulting in an inevitable accumulation of error over time. [Lu and Milios, 1997a,b] addressed this problem by requesting consistency over the entire set of scans, and referred to it as *consistent pose estimation*. They assume that for typical mobile robotics application, there is usually a high degree of overlap among arbitrary scan pairs, since the range of a laser scanner is large compared to the robot's

environment, and the robot traverses areas over and over again. Hence the map-building problem can be formalized as an optimization problem, i.e. maximizing the congruence of set of scans with the individually associated poses as the parameters. It is, however, difficult to find this global maximum. For example, if the optimal match for N=1000 scans has to be found, the corresponding search space is 3N = 3000-dimensional. Lu and Milios use hill-climbing to find this optimum, but since hill-climbing can easily be trapped in a local minimum, it is only suitable for small corrections of good initial position estimates. If a robot moves in large cyclic environment, and traverses a long distance without overlap to previous scans, there can be a substantial difference between its estimated and actual pose when the robot returns to its starting location. This difference can be so large that the robot does not even notice that it closed the loop and reached an already familiar area, and continues the map building process by adding more and more "new" features.

[Gutmann and Konolige, 1999] extend Lu and Milios' work to continuously correlate entire map patches in a background process, in order to detect overlaps and close potentially undiscovered loops. Their approach is incremental and therefore suitable for real-time map generation. [Thrun et al., 1998b] require some manual intervention during the map building process: A button has to be pressed each time the robot passes a "critical" area such as a crossing or a door, thus creating an implicit topological model and limiting the search space for closing the loop. Then, they use the Expectation-Maximization (EM) algorithm, originally proposed by [Dempster et al., 1977], to find the pose parameter configuration for the optimal alignment of the scans. They show that with this method, it is possible to offline correct erroneous tracking to a great extend, and to create a reasonable map even in presence of large odometry errors. However, if the size of the covered area exceeds the range of the laser scanner substantially, arbitrary scan pairs do in general not overlap, and the absolute accuracy of the resulting map is poor for all of the above approaches. Hence, if a global a-priori map is available, it is advantageous to use this map rather than to determine global pose based on cross-consistency.

While applicable for an indoor environment in absence of an initial map, consistent pose estimation would not achieve satisfactory results for a city environment. First, a city is an extremely circular environment, and hence generally difficult for a fully automated consistent pose approach as pointed out above. Second, in contrast to typical indoor robotics applications, scan range is small compared to traveled distances before closing a loop, and arbitrary scan pairs do not generally overlap. Loops can be extremely large, i.e. there are for long driving periods no cross connections to other parts of the map, resulting in extensive position uncertainty for these sections. Third, a city is orders of magnitude larger than an indoor environment. The above algorithms are not capable of handling the simultaneous processing of tens or hundreds of thousands scans. For Lu and Milios' approach, the amount of computations grows $O(n^3)$ with the number n of scans; Gutmann and Konolige as well as Thrun et al. have to subsample the amount of used scans to a few hundreds in order to apply their algorithms, even though the dimensions of their area are

less than a few tens of meters. For these reasons, we cannot rely on scan consistency to globally correct the relative path estimates; we need a global map of the city.

In [Frueh and Zakhor, 2001], we propose the usage of an aerial photo as a global map and to determine the acquisition vehicle's pose in respect to this map. In this thesis we extend the idea to alternatively using a Digital Surface Model (DSM) as a global reference. The basic idea behind our pose correction is that objects visible during ground-based data acquisition must in principle also appear in the airborne view. Making the assumption that the position of building facades and building footprints are identical or at least in most cases sufficiently similar, one can expect that the shape of the horizontal laser scans match edges in an airborne image or a DSM. In the following, we will create an airborne edge map and describe two methods to match the ground based laser scans directly with the map, in order to arrive at global position estimates and hence globally consistent 3D models. The first method is non-probabilistic and needs additionally a digital road map of the city district; the second method is probabilistic, robust, and requires only an aerial photo or an airborne DSM.

5.2 Global Maps from Aerial Images or Airborne Laser Scans

As already stated, even the accuracy of DGPS devices can be as low as meter range in "urban canyons", with frequent signal dropouts. This comes at a price that would more than double the total costs of our entire acquisition system. However, for most urban areas, there are perspective corrected aerial photos and digital roadmaps available, often at no costs. Their resolution is up to sub-meter range, and they provide a geometrically correct view over the entire city area. As an alternative, Digital Surface Models (DSM), which are mainly created from airborne laser scans and provide also a metric view over an entire area, have become increasingly available during the last years. As such, it is conceivable to use either one as a map in order to arrive at global position without use of GPS devices. One more advantage of using aerial photos or a DSM over GPS is that the same airborne data can potentially also be used to derive an airborne 3D model of a city, which can finally be merged with 3D facade models obtained from ground level laser scans. Indeed, in Chapter 7, we describe an approach to merging an airborne surface mesh obtained from a DSM with facade models obtained from ground-based laser scanning. Using the localization methods described in this chapter, the global pose can be determined in respect to the airborne data, thus providing an elegant solution to the registration problem occurring in any model fusion process.

In this section, we create a global edge map representing the footprints of the buildings from aerial photos or from a DSM obtained from airborne laser scans. This map does not contain explicitly defined line segments; rather, we differentiate the image and obtain an *edge image*, in which the intensity value of a pixel is proportional to the local strength of an edge. Since the map area is quantized into discreet pixels, it resembles an occupancy grid representation for discontinuities.

5.2.1 Edge Map from Aerial Photo

In general, photos are created by a perspective projection and as such not metric maps. Only for the special case that the photo is taken from infinity and exactly perpendicular to the ground, the projection becomes parallel and the photo is a metric map of the area; in this case the photo is called ortho-photo. In reality, aerial photos are not taken from infinity, however, if they are taken sufficiently perpendicular to the ground, it is at least approximately metric within any plane parallel to the ground; in particular, the ground is mapped metrically if the terrain is completely flat. All objects outside a flat ground plane have a different metric and the perspective shifts their location in the photo. Unfortunately, the desired footprints of buildings are rarely visible from airborne view; instead, the rooftops are visible. These rooftops are outside the ground plane and show therefore a perspective shift, with a direction depending on their location in respect to the camera's axis of view, and a length proportional to the height of the building. This is shown as an example for the 92–meter Sather Tower on the Berkeley campus in Figure 5-2. Note that for the surrounding lower buildings the perspective shift is far less noticeable.

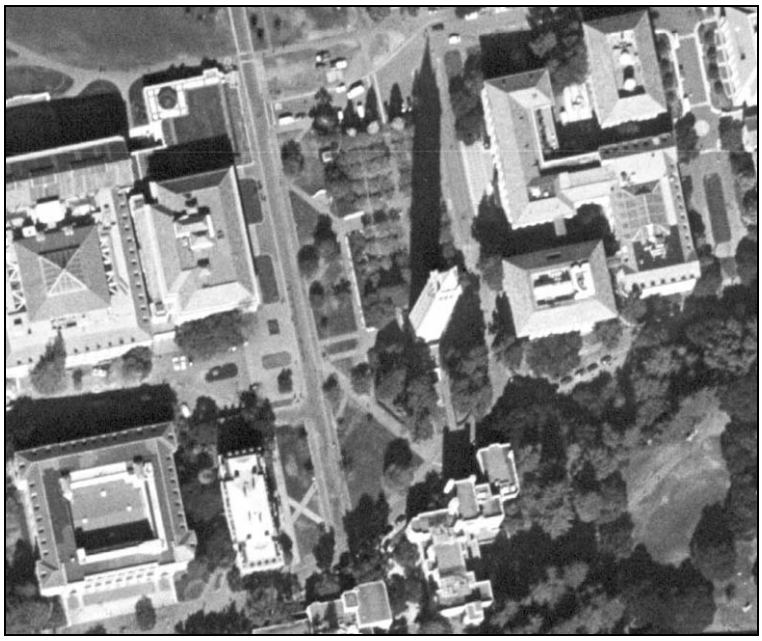


Figure 5-2: Perspective shift for the 92-meter Berkeley campanile (Sather Tower) and surrounding lower buildings

The farther the distance between camera and ground, the less the perspective shift. In case of infinite distance, the perspective camera projection turns into a parallel projection; there is not perspective shift at all and the photo is perfectly orthographic. Perspective shift can be a crucial problem for areas with many high-rising skyscrapers; however, for the region we are processing, the percentage of tall buildings is sufficiently small. In addition, the algorithms introduced in the next sections are designed to be rather

insensitive to small errors in perspective displacement, and we will show that it is possible to obtain a consistent model. However, for some cities this assumption can eventually not be made. One possibility is then to use high-resolution satellite image instead of aerial photos, since they are taken from a much higher distance to the ground. A second possibility is to determine the height of buildings by stereo vision algorithms, and to use the obtained disparity map to correct for perspective shifts and create an artificially orthographic image.

Besides the shift of rooftops, there are other sources of errors: Edges in the image are not only originate from buildings, but also from false, non-3D objects such as road stripes, crosswalk borders, and curbs. Especially problematic are shadows, because they typically result in strong edges in the image, as seen in Figure 5-2 for the campanile. For these edges, there is no corresponding 3D object visible in the ground-based scans, hence reducing the similarity of scans and edge map. Additionally, aerial photos and ground-based scans may not have been taken at the same time, so that their content is potentially different. For our data, we have noticed that some places have been entirely changed between the two acquisitions, for example a former parking lot has been replaced by a building. Also, dynamic objects such as cars or buses can cause discrepancies.

Despite these potential problems, we assume that for our data set perspective is negligible and that on average the actual, correct 3D edges are dominating. Assuming that there are intensity boundaries between different objects such as buildings and ground, we can detect object boundaries as edges in the image. We do not want explicitly extract contiguous edges or compute a polygonal representation; rather, we intend to incorporate all information contained in the photo equally in the edge image. For our purpose, a simple Sobel edge detector is therefore more appropriate than e.g. a sophisticated Canny edge detector, which tries to track intensity discontinuities. For example in tree areas, the Canny detector would focus on the sharp boundaries of the shadow, while omitting the weaker actual tree boundaries. For the same reason, we do not apply common image processing steps such as thresholding after the differentiation. Applying a Sobel edge detector to an aerial image, we obtain an edge image, where the intensity of a pixel is proportional to the strength of its 3D discontinuity. Figure 5-3(b) shows an aerial image from downtown Berkeley and the corresponding edge image.

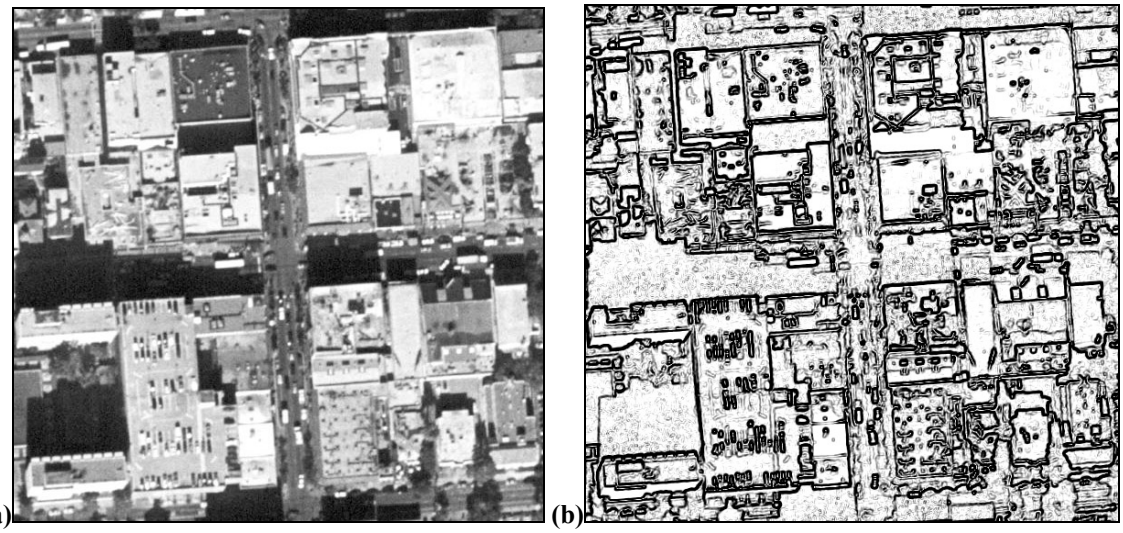


Figure 5-3: Edge map from aerial photo; (a) original aerial photo, and (b) edge map obtained after Sobel filter

5.2.2 Edge Map from DSM

An alternative source of a global edge map is a Digital Surface Model (DSM), which is an array of altitude points uniformly sampled over a 2D area, and can as such be regarded as an airborne height map or depth image. A DSM can be obtained by manual or automated matching of stereo images, by Synthetic Aperture RADAR (SAR), or from airborne laser scans. While we do not make any restrictions about the source of the DSM, we will in Chapter 7 specifically go into details as how to create a DSM from airborne laser scans. Figure 5-4 shows an example for a DSM visualized as a depth image. Each pixel represents a 0.5 by 0.5 meter area on the ground, and its gray value is proportional to the altitude at this location. Since each DSM implicitly defines a depth image, we will in the following use the term DSM also to describe the associated image.

A DSM is a better source for a global edge map than aerial images, for the following reasons: First, it contains two different types of information usable for localization of the data acquisition vehicle, (a) the location of building facades as height discontinuities, and (b) the terrain shape and hence the altitude z of the streets the vehicle is driven at. Second, all intensity differences in the DSM are actual 3D discontinuities, whereas for aerial photos, high intensity differences for shadows of buildings or trees result in false edges. Third, there is no perspective shift of building tops; the DSM is virtually a perfect ortho-image. The scan points forming the basis of the map are given in their true world coordinates, and as described in Chapter 7, the map is in fact constructed by a parallel projection perpendicular to the ground. Therefore, in contrast to the perspective projection of the aerial photo, all image edges are independent of the building height at the correct x,y location in the ground plane. However, one problem remains for both images: there can be a difference between the roof of a building and its footprint, i.e. its shape in the horizontal plane in which the ground-based scans are taken, if for example the building has an overhanging roof.

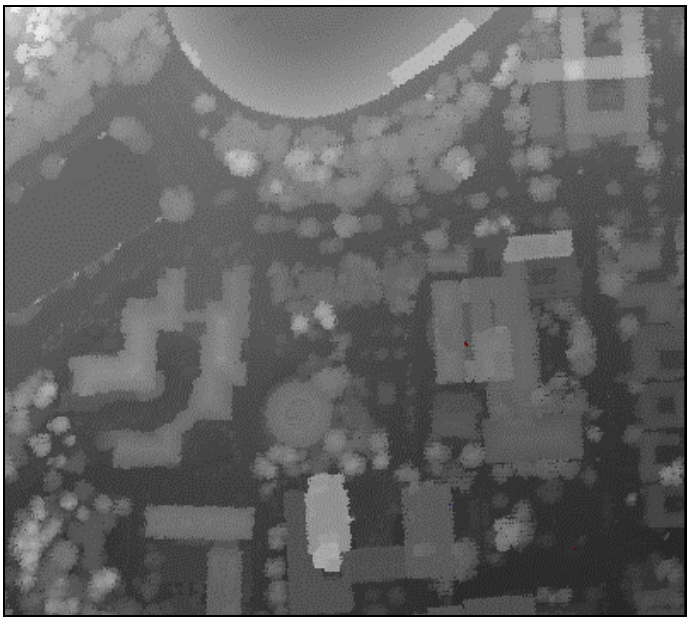


Figure 5-4: Digital Surface Model, displayed as a depth image

To detect edges in the DSM, we could in principle apply again a Sobel edge detector directly to the gray level representation of the DSM. However, the Sobel edge detector treats the darker and the brighter sides of an image discontinuity equally and marks the boundary pixels at both sides. Since the width of an edge is two pixels even at a sharp discontinuity, it tends to produce thick edges. For a DSM, we have more information at discontinuities: while in an aerial photo one cannot make reliable assumptions about which side of a discontinuity is the building top and which is the ground, this information is explicitly given in a DSM. Instead of the Sobel edge detector, we define a discontinuity detection filter, which marks a pixel if at least one of its neighboring pixels is more than a threshold Δz_{edge} below it, i.e

Discontinuity edge filter:

```
for x:=0 to dsm_dimensionX  {
    for y:=0 to dsm_dimensionY  {
        is_edge(x,y) := false;

        for all neighbors of (x,y) {
            if z(neighbor) <  z(x,y)- Δz_edge then {
              is_edge(x,y) :=  true;
            };
        };
    };
};
```

Hence, only the outermost pixels of taller objects such as building tops are marked, but not the adjacent ground, and the resulting edge map is sharper than the edge map obtained from a Sobel filter. It is in fact a global occupancy grid for building walls. Figure 5-5(a) shows an example of an edge map resulting from a Sobel filter, Figure 5-5(b) shows the results of the proposed discontinuity filter for comparison, applied to the DSM of Figure 5-4.

(a) (b)

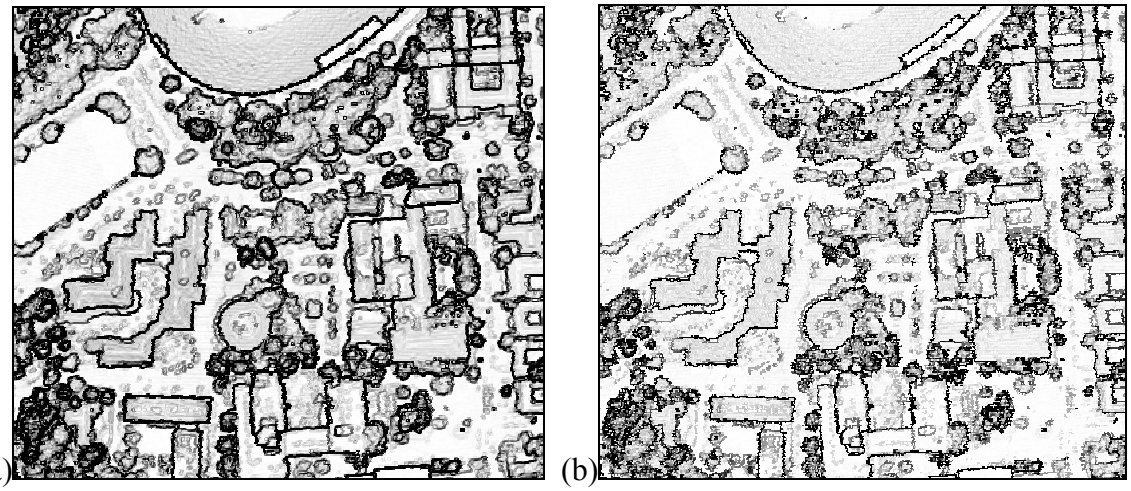


Figure 5-5: Edge map obtained from DSM with (a) Sobel filter, (b) proposed alternative discontinuity filter

The second type of information in the DSM is the terrain shape. Since our vehicle always drives on the road, an estimate for its z coordinate can be obtained from the terrain altitude. It is not possible to directly use the z value at a DSM location, since the airborne laser captures cars on the road and overhanging trees during our airborne data acquisition, resulting in z values up to several meters above street level at these locations. For a particular location, we estimate the altitude of the street level by averaging over the z coordinates of available ground pixels within a surrounding window, weighing them with an exponential function decreasing with distance. The result is a smooth, dense Digital Terrain Model (DTM), containing ground level estimates for the terrain shape for roads as shown in Figure 5-6(b). The terrain altitude at building locations is rather hypothetical and not of any interest for our application, since the vehicle has certainly not been at these locations.

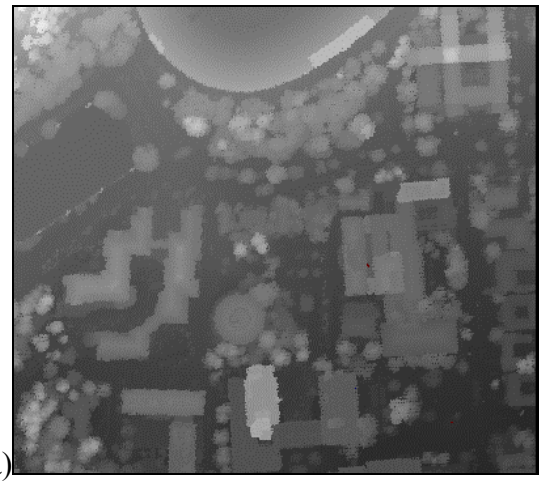
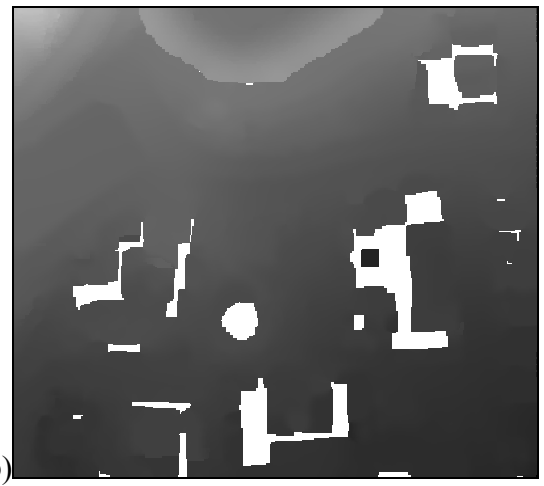
(a)  (b) 

Figure 5-6: (a) Original DSM, (b) estimated DTM, with some blank spots at building locations

5.3 Congruence Coefficient between Ground Based Laser Scans and Airborne Edge Maps

In this section, we compute a measure as how well the ground-based laser scan points match to the airborne edge map. Given a 2D pose (x,y,θ) of the truck in the world coordinate system and the corresponding horizontal laser scan, we can transform the local coordinates (u_j,v_j) of the j^{th} scan point into edge map coordinates (x'_j,y'_j) according to

$$\begin{pmatrix} x'_j \\ y'_j \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + R(\theta) \cdot \begin{pmatrix} u_j \\ v_j \end{pmatrix} \tag{5-1}$$

where R(θ) is the 2 x 2 rotation matrix with angle θ. Summing up the intensity values of the corresponding pixels, we define a coefficient c(x,y,θ) for the congruence between edge image and scan as

$$c(x,y,\theta) = \frac{\sum_j I(x'_j, x'_j)}{\sum_j I_{\max}}, \tag{5-2}$$

where I(x,y) denotes the intensity of the edge image at the world coordinates (x,y) and I_{max} its maximum possible value. The division by the maximum value normalizes c(x,y,θ) to the range [0, 1], with c(x,y,θ) = 1 if all scan points are at an edge of maximum strength, i.e. perfect match, and c(x,y,θ) = 0 if no scan point falls on an edge, i.e. no match at all. One can regard the ensemble of laser scan points in the local coordinate system as a second edge image; in this view, equation 2 essentially computes the image correlation between the two edge images as a function of translation and rotation. Therefore, we can refer to c(x,y,θ) also as a cross correlation coefficient. For the same scan, different global

position parameters (x,y,θ) yield different coefficients $c(x,y,\theta)$; the highest coefficient is obtained for the parameter set for which the scan and the edge map match best.

Figure 5-7 shows two examples for the congruence coefficient; for each figure, the laser scan points (black) are superimposed on the edge image (gray), and the hypothesized position (x,y) is marked as an encircled cross. The scan corresponding to (x_a, y_a, θ_a) shown in Figure 5-7a has a coefficient of only $c(x_a, y_a, \theta_a) = 0.377$, whereas the maximum congruence, 0.527, occurs at (x_b, y_b, θ_b). In this example, the difference between the two pose parameter sets is

$$(\Delta x_a, \Delta y_a, \Delta\theta_a) = (x_a, y_a, \theta_a) - (x_b, y_b, \theta_b) = (4.0\text{ m}, 4.5\text{m}, -2^o),$$

denoted by the arrow shown in Figure 5-7b. While at a first glance, the difference between the two coefficient values does not seem to be striking, the difference is actually significant and the maximum sharp. As seen, in accordance to the coefficient, the scan points for pose (x_a, y_a, θ_a) fit the edges significantly more closely than the ones for pose (x_b, y_b, θ_b).

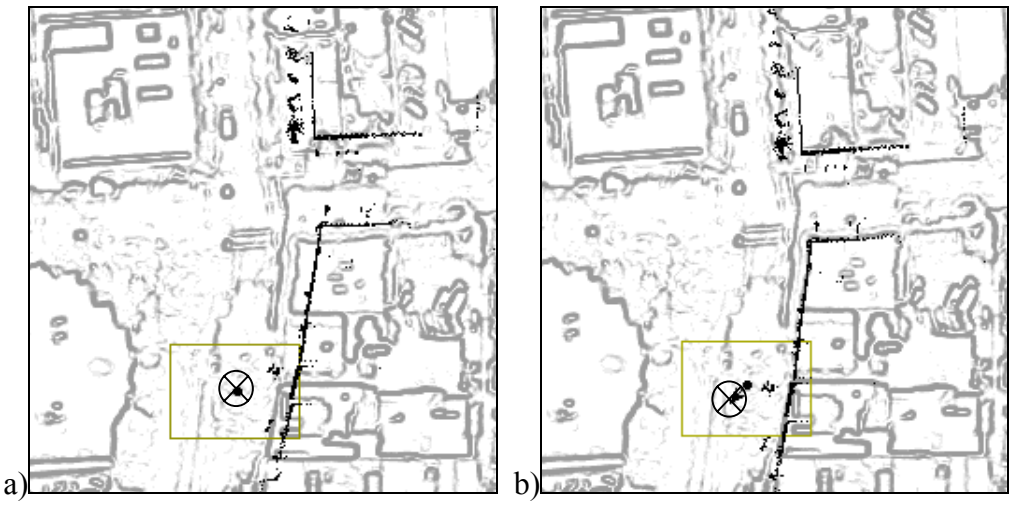


Figure 5-7: Edge image with (a) scan superimposed from pose (x_a, y_a, θ_a), with $c(x_a, y_a, \theta_a) = 0.377$; (b) at pose (x_b, y_b, θ_b), with $c(x_b, y_b, \theta_b) = 0.527$, optimally matching the airborne edge map

5.4 Global Map Position by Maximizing Congruence

An intuitive method for correcting small errors in the vehicle's global pose is to optimize the congruence coefficient introduced in the previous subsection. Initial pose estimates are obtained by the scan matching and path computation process. Then, the congruence coefficient is computed for a certain parameter range around the initial pose estimate, for example the marked rectangular area $\{\pm\Delta x_{max}, \pm\Delta y_{max}\}$ in Figure 5-7 and various angles. The parameters of the largest coefficient are considered as the true pose, and the path is corrected accordingly.

Unfortunately, as seen in Figure 5-1, the initial path has extreme deviations from the true positions, and at some points substantial corrections are required. While the correct position can be found for most parts without problems, there are some map regions, such as residential areas or parking lots, without clear edge features; even worse, the accuracy of the scan matching and hence of the initial path is especially low for these regions, as explained in Section 4.2. Once the deviation from the true pose exceeds the search window around the initial pose estimate, the true pose cannot be found any more and the track of the path is entirely lost. Therefore, it is not possible to recover the entire driven path with this procedure alone.

We suggest two different solutions for this problem:
(a) Applying a coarse adjustment to the initial path by using digital roadmaps in order to bound the deviations, and "fine-tuning" the resulting adjusted path by maximizing congruence; or (b), using more sophisticated, robust probabilistic localization methods to consider the increasing uncertainty in difficult regions.

5.4.1 Adjustment Using Digital Roadmaps

Digital roadmaps are available for all major cities worldwide; in particular, for the Bay Area in California, roadmaps registered with aerial images can be downloaded from the United States Geographic Survey (USGS)'s web sites. A roadmap can be interpreted as a graph, where every intersection or turn is a node, and the road segment in between is an edge. The important information the digital roadmap can provide is the topology and geometry of the city, and hence the possible driving paths. In this sense, our approach is similar to the one in [Thrun et al., 1998b], which also employs a topological map to reduce search space, but in our approach there is no manual step involved. If for example during a vehicle turn operation an overall estimation error of several degrees occurs, roadmaps can be used to correct the angle. The accuracy of this method is limited to the width of the road, which we assume to be unknown. It is especially important that we recover the "high-frequency" components of the vehicle's path, e.g. lane changes, because for these cases the resulting 3D model would have incorrect shapes if the driving path is assumed to be a straight line.

In order to find and assign road segments to the traveled path, we make the following assumptions:

1. *The starting position is on a road node.*
2. *The truck can only move along the roads in the map and never off road.*
3. *Significant changes in driving direction (turns) necessarily occur on road nodes and nowhere else.*

Note that in a digital roadmap, a curvy road without cross intersections is represented as a sequence of nodes with only one possible connection at an angle.

Given the path $\{(x_k, y_k, \theta_k)\}$ computed from the relative position changes, a line segment approximation of our traveled path as shown in Figure 5-8 can be obtained by detecting all turns as major changes in driving directions, and fitting a straight line segment between two turns. Each line l_i has a corresponding driving vector d_i; the intersection of neighboring lines results in nodes. Since intermediate nodes may be passed without changing driving direction significantly, we apply a tree search to the digital roadmap to find the node where direction and traveled distance fits best to the approximation.

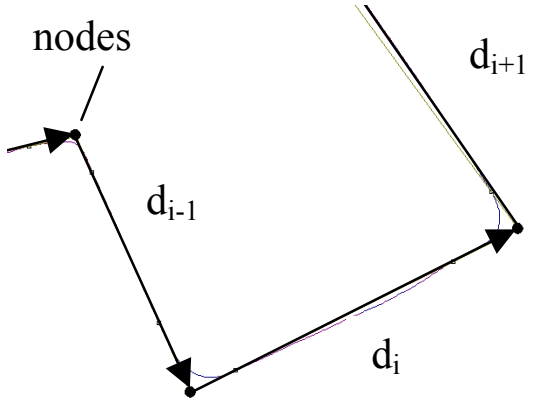


Figure 5-8: Path and its line segment approximation

Our proposed algorithm to track position on the roadmap can be summarized this way:

1) *Select manually the staring point S of the traveled path on the map.*
2) *Choose the first driving vector d_1 in the line segment approximation and add it to S in order to determine the goal point G on the map:*
3) *Start a tree search in the roadmap that finds all possible road paths that have approximately the same direction as d_1 and no major direction changes between successive road edges.*
4) *Compute the Euclidean distance of each road node N_k passed in 3) to the goal point G and find the one with the shortest distance $N_{opt.}$ This is the most probable end node and as such a correction angle $\Delta\theta$ and a length correction factor η can be computed.*
5) *Stretch the direction vector d_1 by η.*
6) *Rotate all d_j with $j \geq 1$ by the correction angle $\Delta\theta$.*
7) *Take G as new starting point S.*
8) *Repeat steps 1 through 7 for all line segments l_i.*

Using this algorithm we obtain a vector graph on the road map and therefore a correction for the length and the angle of each line segment. These adjustments are applied to the initial position estimates $\{(x_k, y_k, \theta_k)\}$, while taking into account the quality value Q_k computed for each step k during the scan-to-scan match described in section 4.1. We

assume that length errors occur mainly during long straight paths, whereas orientation errors are mainly made during turns, and hence we distribute corrections accordingly, weighing them inversely proportional to Q_k. The result is an adjusted path estimate $\{(x_k', y_k', \theta_k')\}$ that fits approximately to the roadmap, while it is still ambiguous to within the width of the road, i.e. several meters. Also, the path is only bound to the road map at intersections, but not between them, and can therefore deviate more than the width of the road only. This becomes particularly noticeable when the acquisition vehicle travels a long distance in between scanning of the two sides of a given road. However, the remaining deviation is now small enough so that the next step, maximizing the congruence coefficient in order to refine the correction, can be applied.

The adjustment procedure and its result are shown in Figure 5-9. In Figure 5-9(a), the initial path estimate obtained from scan-to-scan matching is drawn, overlaid on top of the digital roadmap. While the basic shape of the driven path is clearly visible, the absolute position is increasingly incorrect after a few hundred meters, mainly due to angle errors in turns. Applying the tree search in the road map, we find the traveled roads as a vector graph, containing the recovered sequence of traversed nodes, i.e. road intersections, as marked in the roadmap shown in Figure 5-9(b). Finally, Figure 5-9(c) shows the resulting adjusted path after distributing the corrections among the relative estimates. The shape of this path matches the actual roads significantly better, especially at node points. However, between nodes it is sometimes more inaccurate than road width alone could explain, mainly because our adjustment distributes orientation errors only in turns and length errors only on straight stretches, which is not necessarily where the errors actually occur.

(a) Initial path estimate from scan-to-scan matching

(b) Vector graph

(c) Path after correction

Figure 5-9: Initial path, vector graph, and corrected path superimposed on the digital road map (gray).

5.4.2 Pose Refinement Based on Maximizing the Congruence Coefficient

To obtain estimates for the absolute map poses $(\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$ with the maximum cross correlation between edge map and scan, we assume that the adjusted relative estimates $\{(\Delta u_k', \Delta v_k', \Delta \varphi_k')\}$ computed in the previous section are close enough to the actual values, and therefore, we iteratively apply a relative step and search the parameter space within a search window $(\pm\Delta x_{max}, \pm\Delta y_{max}, \pm\Delta\theta_{max})$ around $\{(x_k', y_k', \theta_k')\}$. The window dimensions $2\Delta x_{max}$ and $2\Delta y_{max}$ are chosen based on the assumed maximum deviation; they depend not only on road width, but include additional heuristics about occurring offsets. As both scan points and edge map are highly discontinuous, the scan-to-map congruence can have completely different values for offsets as small as one or two pixels. Therefore, there are numerous local maximums within the search window, and it is not possible to apply hill-climbing methods to find the global maximum. Fortunately, the parameter space is only three-dimensional and the search window is relatively small, and hence it is feasible to search for the global maximum by sampling the parameter space and apply a hill-climbing search only around the best parameter sample:

Find poses $(\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$ with maximum correlation
{
(x_0', y_0', θ_0') = *selected starting pose in edge map;*

for each relative step k do {
pose = $(x_{k-1}', y_{k-1}', \theta_{k-1}')$ + *(Rot*$(\theta_{k-1}')\cdot(\Delta u_k', \Delta v_k'), \Delta\varphi_k')$;

max_congruence_pose = pose;

/ coarse search – sampling parameter space */*
for dx:= -Δx_{max} *to* Δx_{max} *step* Δx_{coarse}
for dy:= - Δy_{max} *to* Δy_{max} *step* Δy_{coarse}
for dθ:= - $\Delta\theta_{max}$ *to* $\Delta\theta_{max}$ *step* $\Delta\theta_{coarse}$
if c(pose + (dx, dy, dθ)) > c(max_congruence_pose)
max_congruence_pose = pose + (dx, dy, dθ);

/ now refine fine search with steepest decent */*

pose = max_congruence_pose;

do {
last_pose = pose;

if c(pose + $(\Delta x_{fine}, 0, 0)$ *) > c(pose) pose = pose +* $(\Delta x_{fine}, 0, 0)$;
if c(pose - $(\Delta x_{fine}, 0, 0)$ *) > c(pose) pose = pose -* $(\Delta x_{fine}, 0, 0)$;

if c(pose + $(0, \Delta y_{fine}, 0, 0)$ *) > c(pose) pose = pose +* $(0, \Delta y_{fine}, 0, 0)$;

if c(pose - (0, Δy_{fine},0, 0)) > c(pose) pose = pose - (0, Δy_{fine},0, 0);

if c(pose + (0, 0, $\Delta\theta_{fine}$,)) > c(pose) pose = pose + (0, 0, $\Delta\theta_{fine}$,);
if c(pose - (0, 0, $\Delta\theta_{fine}$,)) > c(pose) pose = pose - (0, 0, $\Delta\theta_{fine}$,);

} while (pose != last_pose);

($\hat{x}_k$, $\hat{y}_k$, $\hat{\theta}_k$) = pose;
} next k
}

Applying this method, a series of intermediate global poses ($\hat{x}_k$, $\hat{y}_k$, $\hat{\theta}_k$) is obtained, for witch the congruence is maximal. These poses have a resolution of map pixel size, and among them outliers can occur, since there may be mismatches due to shadows, false edges, and perspective shifts. We define an intermediate correction vector as difference between roadmap-adjusted pose and intermediate global pose. Intermediate poses with low congruence coefficient are not considered, since the matching results are likely not to be reliable, and outliers are found and eliminated by median filtering. Averaging over several neighboring correction vectors, we obtain smoothed vectors and can correct the road-map-adjusted path accordingly. In areas with few reliable global poses, the original roadmap-adjusted path is virtually left unchanged.

While it is possible to correct the path reliably in areas with clear building edges, this method has an apparent disadvantage: once the track of the vehicle in the photo is lost and the believed pose is significantly far from the actual path, the algorithm may not recover if the correct match is outside the search window. This potentially occurs in areas such as suburban houses hidden among trees, in which no distinctive line features, but numerous false edges e.g. from tree shadows, are present. In these areas, the accumulated errors can exceed the correlation search range during longer drives. Unfortunately, extending the search window size is not an acceptable solution, since it does not only increase computation time, but also the possibility of finding remote, erroneous scan-to-photo matches in ambiguous situations. Therefore, the selection of heuristic parameters such as search window size and correction vector weight is crucial for the success of the method. Furthermore, this method represents the pose only as one discreet set of parameters; no measure for its uncertainty is incorporated.

5.5 Global Map Position Based on Monte Carlo Localization

In this section, we investigate the use of Monte-Carlo-Localization (MCL) as a more robust way to improve our pose estimation from laser scans. MCL is an approach in probabilistic robotics and a subclass of Markov Localization. In the following section, we give a short overview over the field of probabilistic robotics, and describe our adaptation of the localization techniques to the specific problem of localizing the acquisition vehicle in the city.

5.5.1 Probabilistic Robotics – Background

In probabilistic robotics, the pose estimate π_t for a time t is not only represented by one single set of parameters, but instead by a probability distribution over the parameter space, hence representing uncertainty of the estimation process. Most approaches make the restrictive assumption, that the environment of the robot is static, although practically they often work also in partially dynamic environments. In a static environment, sensor readings depend only on the state, i.e. the pose of the robot, and not on past or future events or measurements. In other words, the robots pose is the only state in the environment, and it is all one needs to know in order predict the sensor measurements. This assumption is generally known as Markov assumption, and the class of localization approaches making this assumption is called Markov Localization [Russell and Norvig, 1995], [Simmons and Koenig, 1995]. In contrast to mobile robotic applications such as an interactive museum guide, the static environment assumption is easily fulfilled in our case. Since the horizontal scanning plane is well above dynamic obstacles such as cars and pedestrians, laser sensor readouts are not affected and depend only on static objects such as trees and buildings.

An excellent overview over the major concepts as well as an extension for dynamic environments is given in [Fox et al., 1999a]. The key idea of Markov localization is to interpret the state, i.e. pose π_t, at the time t as a random variable, and to estimate its probability density, typically called the *belief*, conditioned on a series of input data. Formally, the believe can be denoted as

$$Bel(\pi_t) = p(\pi_t \mid d_t..d_0) \tag{5-3}$$

where $d_0..d_t$ is the input data from time 0 to time t. Input data can be distinguished into *motion* data, denoted as *a* for action and *perception* data, denoted as *o* for observation. For example, motion data is typically odometry information, and perception data is information captured by sensors. In this notation, the data inputs d_t are discretized: an action a_t summarizes all effects from the time interval [t-1; t], and an observation o_t is a perception snapshot at time t. $Bel(\pi_t)$ is the updated belief *after* consideration of d_t; consequentially, the belief remains unchanged until the next data set arrives. As such, only motion changes the *actual* state of the system while the perception does not; however, both affect our belief about the state. Starting from an initial belief $Bel(\pi_0)$, for example a Dirac function if the starting pose is exactly known, or a uniform distribution if unknown, the belief is recursively updated as new data comes in.

It has to be a distinguished whether the incoming data is perception data or motion data:

b) Motion data
In this case, d_t is a_t and Equation 5-3

$$Bel(\pi_t) = p(\pi_t \mid d_t..d_0) = p(\pi_t \mid a_t, d_{t-1}..d_0)\,. \tag{5-4}$$

Using the theorem of Total Probability, this can be written as

$$\begin{aligned} Bel(\pi_t) &= p(\pi_t \mid a_t, d_{t-1}..d_0) \\ &= \int p(\pi_t \mid a_t, d_{t-1}..d_0, \pi_{t-1}') \cdot p(\pi_{t-1}' \mid a_t, d_{t-1}..d_0) \cdot d\pi_{t-1}' \end{aligned} \tag{5-5}$$

Again, Markov assumption suggests for the first term on the right-hand side

$$p(\pi_t \mid a_t, d_{t-1}..d_0, \pi_{t-1}') = p(\pi_t \mid a_t, \pi_{t-1}'), \tag{5-6}$$

and since π_{t-1} does not depend on a_t, the second term can be written as

$$p(\pi_{t-1}' \mid a_t, d_{t-1}..d_0) = p(\pi_{t-1}' \mid d_{t-1}..d_0) = Bel(\pi_{t-1}'), \tag{5-7}$$

hence yielding the final recursive update equation

$$Bel(\pi_t) = \int p(\pi_t \mid a_t, \pi_{t-1}') \cdot Bel(\pi_{t-1}') \cdot d\pi_{t-1}'. \tag{5-8}$$

The term $p(\pi_t \mid a_t, \pi_{t-1}')$ denotes the probability of a state π_t given an action a_t and an previous pose π_{t-1}'; to compute this probability, a motion model is necessary. In the motion step, two probability distributions are convoluted; intuitively, the motion flattens the probability distribution, because additional uncertainty is introduced, as shown in Figure 5-10 for a one-dimensional state space: The initial belief has its peek at x_p, then, an uncertain motion in the x-direction with expected value Δx occurs, shifting and flattening the resulting belief.

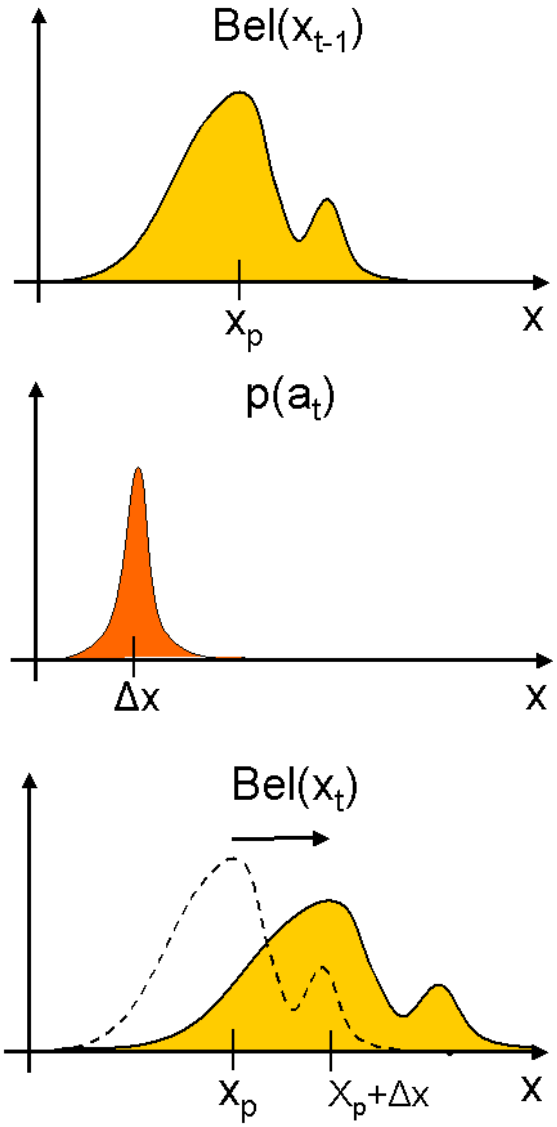


Figure 5-10: Updating the believe for the x coordinate with motion data; (a) initial belief, (b) action estimate with uncertainty, (c) new belief as the convolution if the two probability densities

a) Perception data

In this case, d_t can be written as o_t and the belief can be expressed as

$$Bel(\pi_t) = p(\pi_t \mid d_t..d_0) = p(\pi_t \mid o_t, d_{t-1}..d_0). \qquad (5\text{-}9)$$

Using Bayes' rule, this can be transformed to

$$Bel(\pi_t) = \frac{p(o_t \mid \pi_t, d_{t-1}..d_0) \cdot p(\pi_t \mid d_{t-1}..d_0)}{p(o_t \mid d_{t-1}..d_0)}. \quad (5\text{-}10)$$

Due to the Markov assumption, o_t depends only on the state π_t and not on $d_0..d_{t-1}$, so that

$$p(o_t \mid \pi_t, d_{t-1}..d_0) = p(o_t \mid \pi_t). \quad (5\text{-}11)$$

Since the denominator is a constant in regard of π_t, it can be put in a normalization factor

$$\eta = \frac{1}{p(o_t \mid d_{t-1}..d_0)}, \quad (5\text{-}12)$$

hence simplifying Equation 5-10 to

$$Bel(\pi_t) = \eta \cdot p(o_t \mid \pi_t) \cdot p(\pi_t \mid d_{t-1}..d_0). \quad (5\text{-}13)$$

Furthermore, we can write $p(\pi_t \mid d_{t-1}..d_0) = p(\pi_{t-1} \mid d_{t-1}..d_0)$, since the belief does not change if no additional data arrives, so that Equation 5-13 can finally be written as

$$Bel(\pi_t) = \eta \cdot p(o_t \mid \pi_t) \cdot Bel(\pi_{t-1}), \quad (5\text{-}14)$$

providing a recursive formula to update the belief for incoming sensor data. The normalization factor η can easily be computed with the theorem of Total Probability, i.e.

$$\int_{\infty} Bel(\pi_t)\, d\pi_t = 1 \quad (5\text{-}15)$$

The term $p(o_t \mid \pi_t)$ denotes the probability of the observation o_t given the pose π_t; to compute this probability, a perception model is necessary. In the perception step, two probability distributions are multiplied. Generally, the perception sharpens the position estimate, because additional information is used to modify the distribution, as shown in Figure 5-11 for a one-dimensional state space: The initial belief has its peek at x_p. Then, an uncertain observation suggests that the actual position is slightly to the left, and since the resulting belief is computed by multiplying the two density functions; its maximum is in the "common" area. Hence, the resulting belief is sharper.

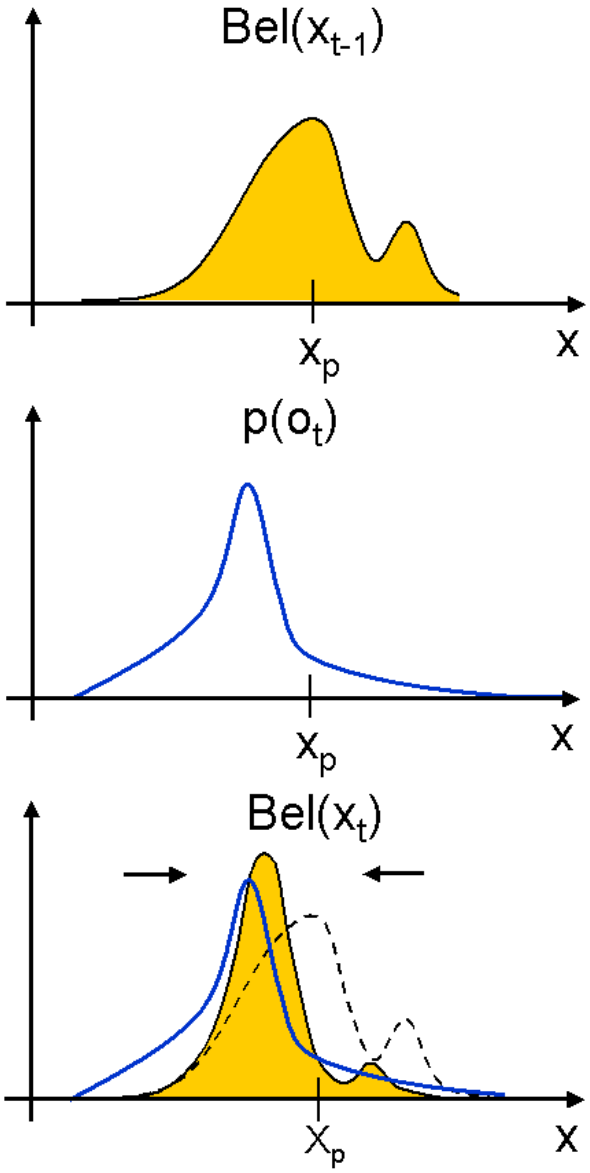


Figure 5-11: Updating the believe for the x coordinate with perception data; (a) initial belief, (b) observation estimate with uncertainty, (c) new belief as the multiplication of the two probability densities

Markov localization provides the recursive update equations for both, motion and perception data; however, it is neither specified how the probability densities can be represented, nor how the probabilities $p(o_t \mid \pi_t)$ and $p(\pi_t \mid a_t, \pi_{t-1}')$ can be obtained. The computation of $p(\pi_t \mid a_t, \pi_{t-1}')$ is usually easier, since the motion model for a robot is rather precise; the exact computation of $p(o_t \mid \pi_t)$ is more difficult. Fortunately, it is sufficient to possess only very approximate knowledge of the exact perception and motion probabilities $p(o_t \mid \pi_t)$ and $p(\pi_t \mid a_t, \pi_{t-1}')$. Practically, the results do not differ

substantially, as long as the behavior of modeled and actual probabilities is sufficiently similar.

As this method carries on multiple hypotheses, it is robust enough to recover from position errors and mismatches, and capable of finding the correct pose, even if the initial pose is not given precisely or at all. However, the crucial point in Markov localization is the implementation, because a reasonable representation for the probability distribution needs to be found. Common approximations of the belief in the current literature are:

a) Gaussian approximation

Based on the Kalman filter [Kalman, 1960], the restrictive assumption is made that belief, action and observation are Gaussians, and can hence be represented by only two parameters: mean value and variance. Since this assumption renders computation simple and efficient, it has been very popular for tracking a robot real-time. In this context, it has been applied successfully in many applications, and has proven to be robust, if the uncertainty remains small. However, in its plain form, it has the severe drawback that it cannot handle multiple hypotheses in case of ambiguity. This problem has been addressed by proposing multi-hypotheses Kalman filters, which represent the belief as mixtures of Gaussians [Jensfelt and Kristensen, 1999], [Roumeliotis and Bekey, 2000]. However, even multi-hypotheses Kalman filters are not capable of recovering from catastrophic localization failures, such as in the *kidnapped robot* problem or in the *initial global localization* problem. Furthermore, the perception probability $p(o_t \mid \pi_t)$ has also to be assumed as Gaussian, and this is not correct in our case.

b) Grid based representation

A second popular approach is grid-based Markov localization, where the parameter space is sampled as a probability grid, e.g. in [Fox et al., 1999a], [Thrun, 2000]. The parameter space is partitioned into grid cells, each representing the probability in a parameter "cube" by a floating point value. However, for a downtown area, this would lead to more than 10^8 states and hence large computational complexity, even for a resolution as low as 1 meter x 1 meter x 2 degrees.

c) Particle Filtering

In particle filtering [Gordon et al., 1993], also known as *condensation algorithm* in computer vision and as *Monte-Carlo-Localization* (MCL) in the context of robot localization [Fox et al., 2000], a large number of random samples (or particles) is utilized to represent probability distributions. These particles are propagated over time using a combination of sequential importance sampling and resampling steps, shortly referred to as sampling-importance-resampling. The resampling step statistically multiplies or discards particles at each step according to their importance, concentrating particles in regions of high posterior probability. Hence, the particle density is statistically equivalent to the probability density. Although powerful and robust, particle filters are intuitive and relatively simple to implement.

5.5.2 Monte-Carlo-Localization

Due to the large extend of a city environment, the necessity to represent multiple distinct hypothesis, and the requirement to recover from localization failures, the usage of particle filters appears most favorable to us. In our particular MCL problem, π is a three-dimensional state variable with the parameters (x, y, θ), the motion estimates are obtained form the scan-to-scan matching results (Δu_k,Δv_k,$\Delta\varphi_k$) for each step k, as described in Chapter 4, and the perception is given by the congruence coefficient c(x, y, θ) between laser scans and aerial edge map. Applying above ideas, we suggest the following procedure:

We represent the probability distribution of π by a set S of particles P_i, each with an importance factor w_i. In an iterative process, the set S_k of N particles is transformed into another set S_{k+1} of N particles by applying the following three phases: (a) motion; (b) perception, and (c) importance resampling. Each particle P_i is associated with a specific parameter set ($x^{(i)}$, $y^{(i)}$, $\theta^{(i)}$), and the number of particles within a "cube" in the state space around (x, y, θ) is proportional to the probability density at (x, y, θ). Therefore, the histogram over ($x^{(i)}$, $y^{(i)}$, $\theta^{(i)}$) of all particles approximates the probability distribution of (x, y, θ). As such, it is this distribution function of the random variable π =(x,y,θ) that is being propagated from iteration k to k+1 based on the scan-to-scan match in the motion phase, and the scan-to-edge map match in the perception phase.

We assume as a motion model $p(\pi_t \mid a_t, \pi_{t-1}')$ that the distribution of the relative position estimates (Δu_k,Δv_k,$\Delta\varphi_k$) obtained from scan-to-scan matching in Chapter 4 is Gaussian with variances σ_u^2, σ_v^2, σ_φ^2. Note that this refers only to the motion model; it does not imply any Gaussian distribution for the belief. Specifically, in the motion phase, we start with the relative position estimate (Δu_k,Δv_k,$\Delta\varphi_k$), and add to it a white Gaussian random vector to obtain a new random vector, i.e.

$$(\Delta\tilde{u}_k, \Delta\tilde{v}_k, \Delta\tilde{\varphi}_k) = (\Delta u_k, \Delta v_k, \Delta\varphi_k) + (n(\sigma_u), n(\sigma_v), n(\sigma_\varphi)) \tag{5-16}$$

where n(σ) denotes Gaussian white noise with variance σ^2, and σ_u^2, σ_v^2, σ_φ^2 represent scan-to-scan measurement noise variance. Intuitively, one can imagine that for some of the particles, the added noise compensates by chance for the actual measurement error. From the scan point alignment experiments in Chapter 4, we have obtained approximate knowledge about σ_u, σ_v, and σ_φ; however, added noise level should be set higher than the true measurement uncertainty. The reason is that otherwise, in case of an extreme matching error, eventually the correct noise addition is not given to any particle, since it is unlikely to draw a value from the far outskirts of the distribution. Thrun et al. note that MCL can fail if assumed noise level is too small, and have recently suggested Mixture-MCL as an extension that might overcome this problem [Thrun et al., 2001].

According to initial path computation, the parameter set of the i^{th} particle P_i, is transformed to

$$\begin{pmatrix} x^{(i)\prime} \\ y^{(i)\prime} \end{pmatrix} = \begin{pmatrix} x^{(i)} \\ y^{(i)} \end{pmatrix} + R\left(\theta^{(i)} + \Delta\tilde{\varphi}_k\right) \cdot \begin{pmatrix} \Delta\tilde{u}_k \\ \Delta\tilde{v}_k \end{pmatrix} \qquad (5\text{-}17)$$
$$\theta^{(i)\prime} = \theta^{(i)} + \Delta\tilde{\varphi}_k .$$

Intuitively, this means that the amount of movement of each individual particle is drawn from a probability distribution function of the random variable ($\Delta u_k, \Delta v_k, \Delta\varphi_k$). As a result of this phase, particles that share originally the same parameter set are "diffused" after the transformation in Equation 5-17.

For the perception model, we utilize the correlation coefficient $c(x^{(i)\prime}, y^{(i)\prime}, \theta^{(i)\prime})$ between horizontal laser scans and aerial edge map as a measure for the probability $p(o_t \mid \pi_t)$. During the perception phase, for each particle with a new pose $(x^{(i)\prime}, y^{(i)\prime}, \theta^{(i)\prime})$, we set a preliminary importance factor w_i^* to the correlation coefficient $c(x^{(i)\prime}, y^{(i)\prime}, \theta^{(i)\prime})$ between laser scans and aerial photos, according to Equation 5-2. We subsequently normalize w_i^* to obtain the true importance factor w_i as follows:

$$w_i = \frac{w_i^*}{\sum_{particles} w_j^*}, \qquad (5\text{-}18)$$

Since $c(x^{(i)\prime}, y^{(i)\prime}, \theta^{(i)\prime})$ is a measure of how well the current scan matches to a particular vehicle pose $(x^{(i)\prime}, y^{(i)\prime}, \theta^{(i)\prime})$, intuitively, the importance factor w_i determines the likelihood that a particular particle P_i is a good estimate for the actual truck position. As such, the importance factor of each particle is used in the selection phase to compute the set S_{k+1} from set S_k in the following way: a given particle in set S_k is passed along to set S_{k+1} with a probability proportional to its importance factor. We refer to the "surviving" particle in set S_{k+1} as a child, and its corresponding original particle in set S_k as its parent. In this manner, particles with high importance factors are likely to be copied into S_k many times, whereas particles with low importance factors are likely not to be copied at all. Thus, "important" particles become parents of many children. This selection process allows removal of "bad" particles and boosting of "good" particles, resembling a sort of evolution process. This selection phase is also referred to as importance sampling.

Starting with an uniformly distributed set S_0 on the starting position in the aerial image, we apply the above three phases at each step k, in order to arrive at a series of sets S_k. More specifically, our algorithm can be summarized as follows:

Monte Carlo Localization:

1. Distribute randomly an initial set S_0 of N particles around the approximate starting position in the edge map

2. For each relative step k
{
// Motion and Perception
For particles i:=0 to N of S_k
a) Generate and add random noise to motion estimate according to Equ. 5-16
b) Move particle P_i according to Equ. 5-17 by distorted motion estimate $(\Delta\tilde{u}_k, \Delta\tilde{v}_k, \Delta\tilde{\varphi}_k)$
c) Set importance factor w_i^* of P_i to congruence coefficient c(new particle pose)

// Importance resampling
1. Normalize importance factors w_i according to Equ. 5-18

2. Pick particles for the next generation with a probability proportional to their importance factor:

```
for i:=0 to N  {
     r = random number(0, 1)
     for j:=0 to N {
          r: = r - w_i;
          if r<0 then  exit  j-loop;   // if w_j is large, it is more likely that
                                       // the loop is exited at this j
     }
     copy P_j into next generation S_k+1
}
```

Figure 5-12 shows one of resulting S_k as an example, superimposed on the aerial image. This set of particles appears like a blob, with the density the highest at the dark spot in the blob.

Figure 5-12: Set of particle representing the pose belief, superimposed on the aerial image. In this visualization, the probability density is expressed by different colors.

It is quite simple to incorporate additional information such as the digital roadmap or eventual GPS readouts during the belief computation, in order constrain parameter space. For example, since we have the digital roadmap registered with the aerial photo available, we can restrict positions of the particles to within a few-meter-wide strip around roads. This can be done efficiently by marking allowed locations in the edge map as shown in Figure 5-13, where the parameter space is restricted to within the darker 25 meters wide strip around roads. Assigning a zero importance to each "off-road" particle, we can prohibit its selection during the resampling process. Though this is not necessary for obtaining the correct path, it can greatly decrease computation time, because incorrect particles can be removed immediately, and therefore much fewer particles are needed, while the probability distribution near the roads is still represented appropriately.

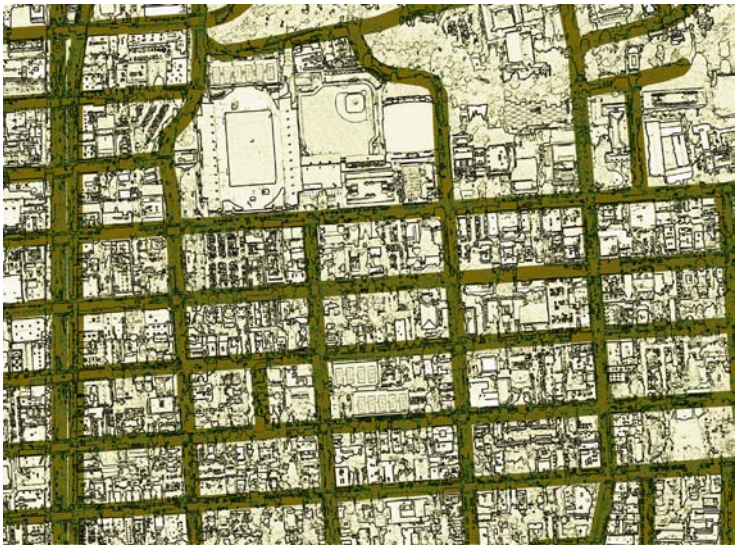


Figure 5-13: Restricting parameter space to locations near roads.

The remaining question is how to obtain the actual global pose at step k from each set S_k, and this is a non-trivial problem. While the belief provides a good measure for the range of possible poses, it is not clear which pose in this range is the actual one. Even if we knew most of the previous and following poses perfectly, we were not able to resolve intermediate uncertainty and determine all locations exactly.

One possibility to obtain a concrete global estimate is to compute the center of mass of the S_k. However, the belief represents only past data, and this would include many particles to be revealed as inappropriate in later steps. For example, if the belief at step k consists of two distinct peeks, one of them to be revealed as erroneous by future data, simply estimating the pose as the center of mass incorrectly results a position in the middle of the two possibilities. Since we know the evolution of the particle sets over time, we can disregard the particles revealed as erroneous in the future. Specifically, we keep track of the "ancestry" of particles and consider only those particles in S_k, whose descendents have survived after M steps later, and hence are in the set S_{k+M}. We then compute the center of mass for these particles in set S_k, and use it as the global 2D map pose estimate for step k. While these estimates are suitable as global references, their absolute position accuracy is not in the centimeter range necessary for an accurate 3D reconstruction. In this sense, they are comparable with GPS readouts, and to obtain the final poses we have to combine the local accuracy of the initial path with the globally correct map positions from MCL.

It would be desirable to adjust the initial path to the global 2D map poses in the same simple and intuitive manner that we used for the congruence maximization, i.e. averaging the correction vectors. However, since MCL does not require an adjustment with a digital roadmap as an intermediate step, we only have the initial path with its strong distortions, and hence extremely large global correction vectors. The essential problem is for the path computation, rotations and translations are coupled, since even small changes in a relative

angle at the beginning of the path can change subsequent positions substantially. One alternative solution is minimizing some distance function between initial path and global points, but since such a function is affected by each parameter of the relative estimates, the resulting optimization problem has some thousand dimensions, and the strong influence of angles for all subsequent positions can cause numerical instability and make the search for the global optimum hard. Fortunately, we can exploit that global yaw angles in the path depend only on previous relative *angle* estimates, but not on *position* estimates. Thus, we can decouple the yaw angle correction from the position correction: we first correct the yaw angles separately by subtracting averaged angle differences between initial and global poses, recompute the path with the new angles, and apply averaged correction vectors to the x and y coordinates. Since this independence is not mutual, it is important that the angle correction is performed *before* the position correction. The result of this correction is a final 2D path, with both local accuracy and correct global pose in all three DOF.

We have assumed perfectly flat environment for global registration in respect to an airborne photo, and hence the 3 DOF (x,y,yaw) describe global pose completely. For the registration in respect to an airborne DSM, we can abandon the restriction to flat environments and create correct models even for hill areas. Utilizing the additional altitude information the airborne laser provides, two more DOF can be estimated in a simple manner: We can fairly assume that the vehicle never leaves the ground while driving in the city. An estimate for the ground level as a smooth 2D manifold has been computed in Section 5.2, and thus we set the final z_k coordinate to the altitude of the ground level at (x_k,y_k) location.

Furthermore, the slope is the gradient of the ground level in driving direction; hence, the slope or equivalently the pitch angle, can be computed as

$$pitch_k = \arctan(slope_k) = \arctan\left(\frac{z_k - z_{k-1}}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}}\right) \qquad (5\text{-}19)$$

by using the altitude difference and the traveled distance between successive positions. Since the resolution of the airborne scans is only about one meter and the ground level was obtained in a smoothing process, the estimated pitch does not contain highly dynamic pitch changes e.g. caused by pavement holes and bumps. Nevertheless, due to its size and length, the truck is relatively stable lengthwise and as such, the obtained pitch is an acceptable estimate.

The last missing DOF, the roll angle, could similarly be estimated using airborne data, but in this case we have two superior alternatives: The first alternative is to assume buildings are generally built vertically, and apply a histogram analysis on the angles between successive vertical scan points. If in average the distribution peak is not centered at 90 degree, the difference between 90 degree and the actual peak angle can be used as roll estimate. The second alternative is to use the ground-based image data to detect the

ups and downs during driving; in contrast to the vertical laser scans, images overlap substantially, thus allowing to determine the relative rolling angles for the camera poses [Flynn, 2002]. Finally, intermediate pose between these dense, accurate global poses are computed by linear interpolation. The result is a 6-DOF path, completely registered with the airborne laser data.

6 Automated Facade Model Generation

In the previous chapters, we managed to solve the problem of accurately determining the vehicle's pose during the data acquisition. In this Chapter, we address the problem of automatically creating a detailed, textured 3D facade mesh from ground-based vertical scans and image data, representing the building walls at the highest level of detail. We propose an ensemble of data processing techniques to create visually appealing facade meshes by removing cluttered foreground objects and filling holes in the building facades. Our objectives are robustness and efficiency with regard to processing time, in order to ensure the scalability to the enormous amount of data for an entire city.

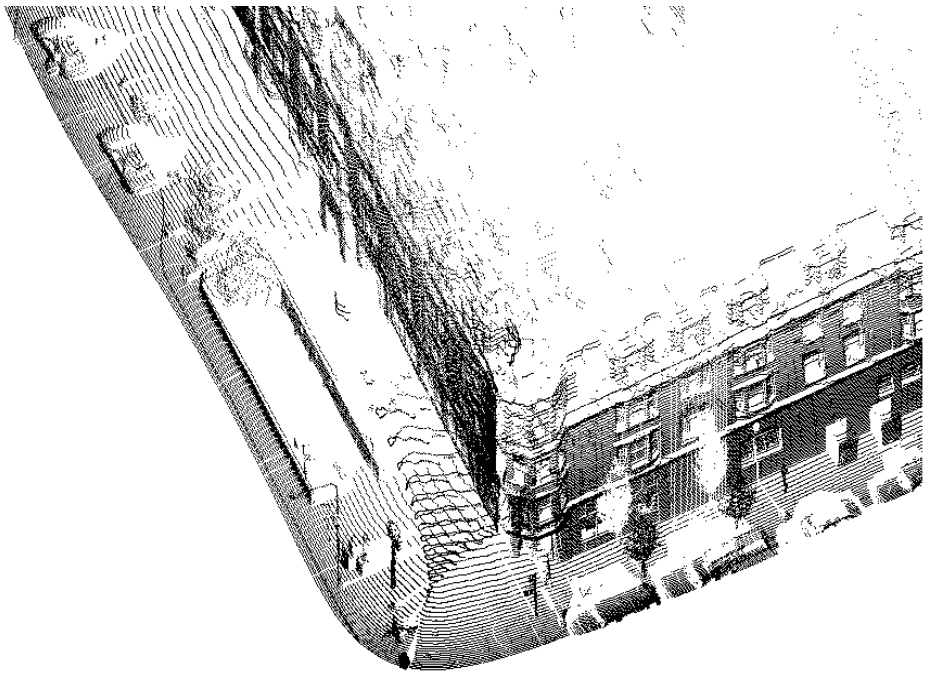


Figure 6-1: Vertical scan points

Knowing the pose of the acquisition sensor accurately, it is straightforward to compute the 3D coordinates of the vertical laser scan points, resulting in a structured point cloud as shown in Figure 6-1. This point cloud contains vertices for any object in the scanner's field of view; many objects in the scene are not desired in a facade model, such as pedestrians, cars and trees. Additionally, there are many erroneous vertices, e.g. due to glass surfaces, and facade areas without any scan point, due to occluding foreground objects. Hence, a simple triangulation of the raw scan points, for example by connecting neighboring points if their distance is below a threshold value, does not result in an acceptable reconstruction of the street scenery, as shown in Figure 6-2(a) and (b). Even though the 3D structure can be easily recognized when viewed from a viewpoint near the original acquisition position as in Figure 6-2(a), the mesh appears cluttered due to several reasons: first, there are holes and erroneous vertices due to reflections off the glass on

windows; second, there are many pieces of geometry "floating in the air", corresponding to partially captured objects and measurement errors; and third, occluding foreground objects such as cars and trees cause large holes in the geometry behind. The mesh appears even worse when viewed from other viewpoints such as the one shown in Figure 6-2(b). Then, the large holes in the building facades caused by occlusion become visible, and furthermore foreground objects become almost unrecognizable when viewed sideways, since the laser scanner has only captured their frontal view. Since we are mainly interested in the building facades, it is our goal to identify foreground and facades in the scans, and to remove foreground objects and fill holes in the facades.

a)

b)

Figure 6-2: Triangulated raw points; (a) front view; (b) side view.

Previous work for scan point processing and model generation has mostly focused on 3D scanners, and thus the general approach to fill gaps caused by occlusions is to combine multiple scans taken from different viewpoints, for example in [Curless and Levoy, 1996], [Stamos and Allen, 2002], [Davis et al., 2002]. In our case scans from other viewpoints are not available, as we drive by a street only once. If we want to fill the holes prohibiting a correct appearance, we rather have to reconstruct occluded areas by only using cues from neighboring scan points, and as such, there has been little work to solve this problem. One approach [Stulp et al., 2001] suggested planar segmentation to 3D laser

scans from an indoor environment, in order to identify foreground objects such as chairs and occluded planar areas behind, e.g. walls. What makes the problem hard in our case of a city environment is the common presence of "uncooperative" materials such as glass and shiny steel, resulting in erroneous scan points, also around the holes. However, in contrast to merged sets of 3D laser scans, our data has the advantage that it comes in a strict row-column fashion, and this regular topology enables the application of fast image processing algorithms. In this Chapter, we describe first our strategy to handle the large amounts of data by a path subsplitting and depth image generation scheme. We will then introduce our algorithms to transform the raw scans into a visually appealing facade mesh, to automatically texture map this mesh, and to create a hierarchy of levels of details to enable interactive rendering.

During data acquisition, we capture simultaneously a long series of vertical and horizontal scans. Using the localization methods in Chapters 4 and 5, the entire "capture" path of the acquisition truck can be reconstructed in a global Cartesian coordinate system [x,y,z], and thus, we can associate an accurate pose estimate with each of the simultaneously captured vertical scans. To partially compensate for the unpredictable, non-uniform motion of the truck, the vertical scan series is subsampled such that the spacing between successive scans is roughly equidistant, e.g. about 10 or 15 centimeters, hence greatly reducing the amount of scans during slow motion or standstill times. This is especially reasonable if one considers the scanner's quite large beam divergency of 15 milliradians, resulting in a spot size and thus a resolution of 15 centimeters in a 10-meter distance, so that denser scans would be completely redundant anyways. The resulting subsampled series of vertical scans S_n is used for the 3D reconstruction. In the following sections, we index the vertical scan by their number n and denote a scan point by its (integer) azimuth angle υ. Furthermore, let $s_{n,\upsilon}$ be the distance measurement on a point in scan S_n with azimuth angle υ. Then, $d_{n,\upsilon}=\cos(\upsilon)\cdot s_{n,\upsilon}$ is the depth value of this point with respect to the scanner, i.e. its orthogonal projection into the ground plane. The scanning setup and its denotations are shown in Figure 6-3.

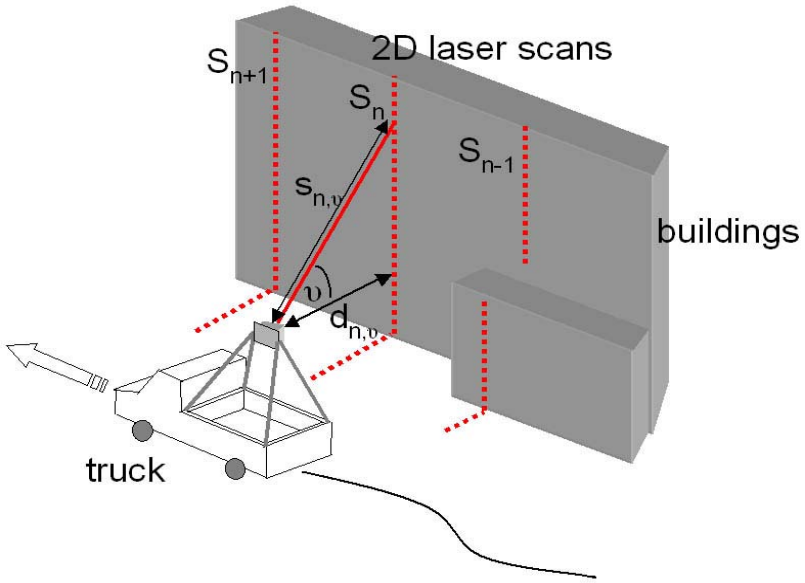


Figure 6-3: Scanning setup and denotations

6.1 Segmentation of the Driving Path Into Quasi-Linear Segments

The data captured during a few-minutes drive consists of tens of thousands of scan columns. Since successive scans in time correspond to spatially close points, e.g. a building or a side of a street block, it is computationally advantageous not to process the entire data as one block, rather to split it into smaller segments to be processed separately. We impose the constraints that (a) path segments have low curvature, and (b) scan columns have a regular grid structure. The latter constrain allows us to readily identify the neighbors to right, left, above and below for each point, and, as seen later, is essential for the generation of a depth image and segmentation operations.

Scan points for each truck position are obtained as we drive by the streets. During straight segments, the spatial order of the 2D scan rows is identical to the temporal order of the scans, forming a regular topology. Unfortunately, this order of scan points can be reversed during turns towards the scanner's side of the car. Figure 6-4(a) and (b) show the scanning setup during such a turn, with scan planes indicated by the two dotted rays. During the two vertical scans, the truck performs not only a translation but also a rotation, making the scanner look slightly backwards during the second scan. If the targeted object is close enough, as shown in Figure 6-4(a), the spatial order of scan points 1 and 2 is still the same as the temporal order of the scans; however, if the object is further away than a critical distance d_{crit}, the spatial order of the two scan points is reversed, as shown in Figure 6-4(b).

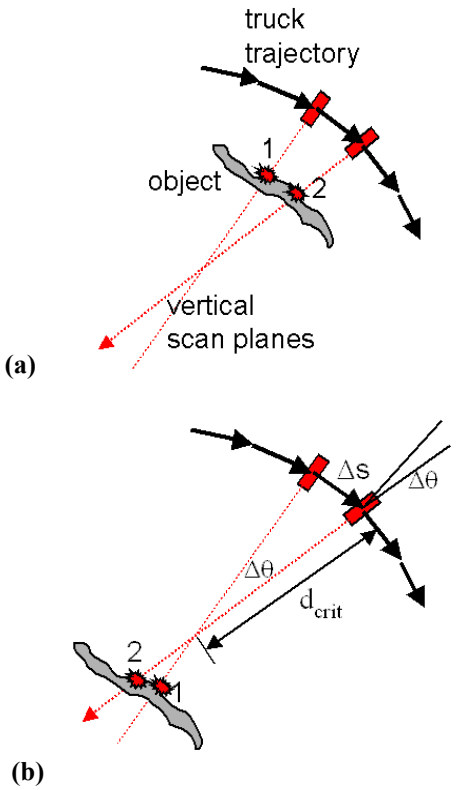


Figure 6-4: Scan geometry during a turn, (a) normal scan order for closer objects; (b) reversed scan order for further objects.

For a given truck translation of Δs and a rotation Δθ between successive scans, the critical distance can be computed as

$$d_{crit} = \frac{\Delta s}{\sin(\Delta\theta)}. \qquad (6\text{-}1)$$

Thus, d_{crit} is the distance at which the second scanning plane intersects with the first scanning plane. For a particular scan point, the order with its predecessors is distorted if its depth $d_{n,v}$ exceeds d_{crit}; this means that its geometric location is somewhere in between points of previous vertical scans. The effect of such order reversal can be seen in the marked area in Figure 6-5, for an acquisition path indicated by the dotted line. At the corner, the ground and the building walls are scanned twice, first from a direct view and then from an oblique angle, and therefore out of order and with lower accuracy. These oblique points destroy the regular topology between neighboring scan points.

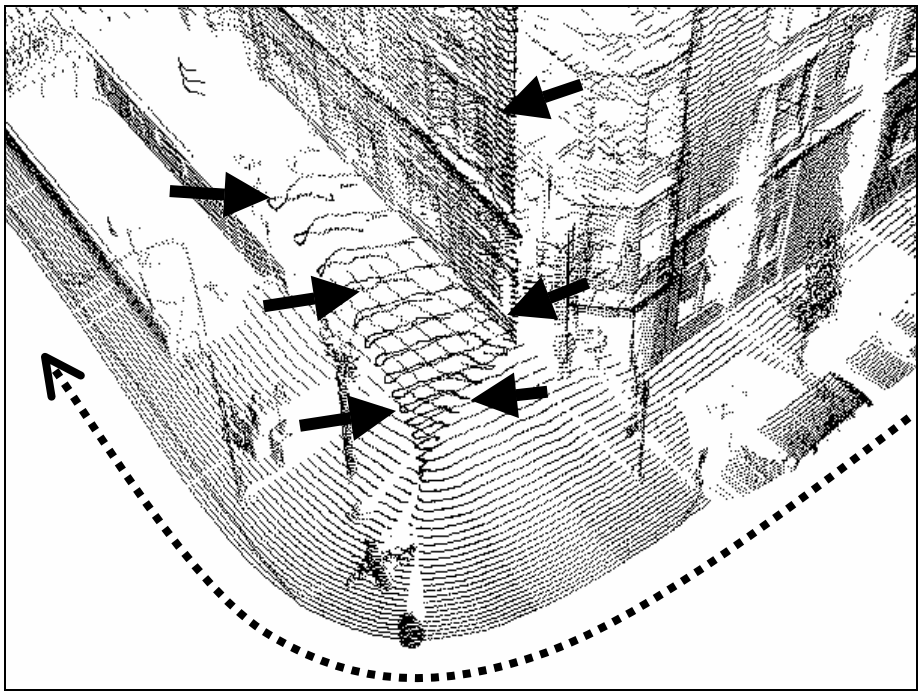


Figure 6-5: Scan points with reversed order at a turn.

Since the "out of order" scans obtained in these scenarios correspond to points that have already been captured by "in order" scans and are therefore redundant, our approach is to discard them and use only the "in order" scans. For typical values of displacement, turning angle, and distance of buildings from our driving path, this occurs only in scans of turns with significant angular changes. By removing these "turn" scans and splitting the path at the "turning points", we obtain path segments with low curvature that can be considered as locally quasi-linear, and can therefore be conveniently processed as depth images, as described in the following section. In addition, to ensure that these segments are not too large for further processing, we subdivide them if they are larger than a certain size. Specifically, in segments that are longer than 100 meters, we identify vertical scans that have the fewest scan points above street level, corresponding to empty regions in space, and divide at these locations. Furthermore, we detect redundant path segments for areas captured multiple times due to multiple drive bys, and use only one of them for reconstruction purposes. Figure 6-6(a) and Figure 6-6(b) show an example of an original path and the resulting path segments, respectively, both overlaid on a roadmap. The small lines perpendicular to the driving path indicate the scanning plane of the vertical scanner for each position.

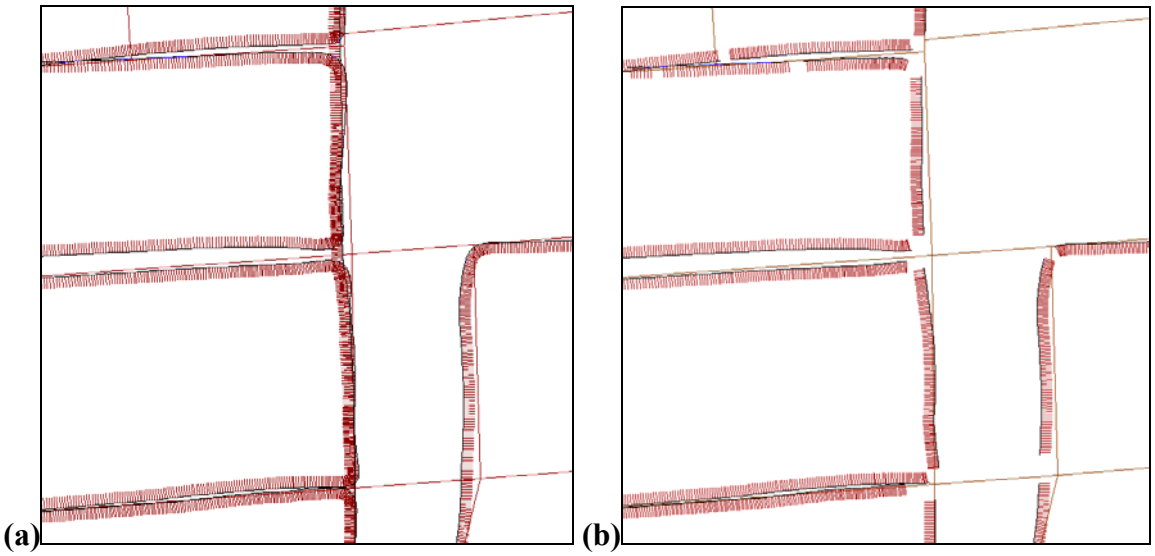


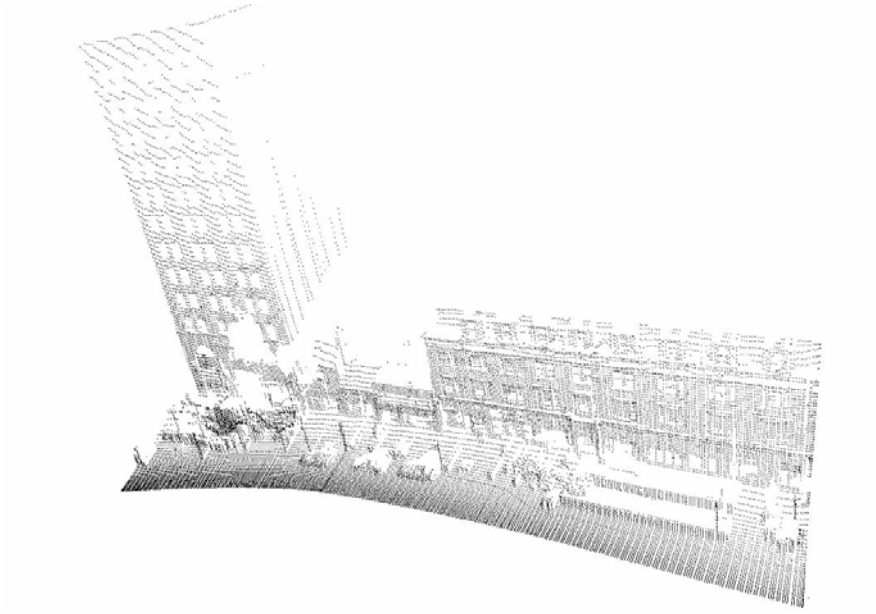
Figure 6-6: Driven path; (a) before segmentation; (b) after segmentation into quasi-linear segments.

6.2 Converting Path Segments To Depth Images

In the previous section, we create path segments that are guaranteed not to contain scan pairs with permuted horizontal order. As the vertical order is inherent to the scan itself, all scan points of a path segment form a 3D scan grid with regular, quadrilateral topology. This 3D scan grid can be transformed into a 2.5D representation, i.e. a depth image in which each pixel represents a scan point, and the gray value for each pixel is proportional to the depth of the scan point. The advantage of a depth image is its intuitively easy interpretation, and the increased processing speed the 2D domain provides. However, most operations that are performed on the depth image can be done as well on the 3D point grid directly, just not as conveniently.

A depth image is typically used for representing the data from 3D scanners. While image size and resolution are dependent on the specific scanner, the depth value assigned to each pixel is usually the distance between scan point and scanner origin, or its cosine with respect to the ground plane. As we expect mainly vertical structures, we choose the latter option and use the depth $d_{n,\upsilon} = \cos(\upsilon) \cdot s_{n,\upsilon}$ rather than the distance $s_{n,\upsilon}$, so that the depth image is basically a tilted height field. The advantage is that in this case points that lie on a vertical line, e.g. a building wall, have the same depth value, and are hence easy to detect and group. Note that our depth image differs from one that would be obtained from a normal 3D scanner, as it does not have one single center from which the scan points are measured. Instead, there are different centers for each individual vertical column along the path segment. The obtained depth image is neither a polar nor a parallel projection; it most resembles to a cylindrical projection. Due to non-uniform driving speed and non-linear driving direction, these centers are in general not on a line, but on an arbitrary shaped, though low-curvature curve, and the spacing between them is not

exactly uniform. Because of this, the grid position specifies in the strict sense only the topological order of the depth pixels, and not the exact 3D point coordinates. However, topology and depth value are a good approximation for the exact 3D coordinates, especially within a small neighborhood. While the depth image facilitates the use of standard image processing techniques such as region growing, the actual 3D vertex coordinates are still kept and used for some 3D operations such as plane fitting. Figure 6-7(a) shows an example of the 3D vertices of a scan grid, and Figure 6-7(b) shows its corresponding depth image, with a gray scale proportional to $d_{n,v}$.



(a)

(b)

Figure 6-7: Scan grid representations; (a) 3D vertices; (b) depth image.

6.3 Properties of City Laser Scans

In this section, we briefly describe special properties of scans taken in a city environment, resulting from the physics of a laser scanner as an active device measuring time-of-flight of light rays. It is essential to understand these properties and the resulting imperfections in distance measurement, since at times they lead to scan points that appear to be in contradiction with human eye perception or a camera. As the goal of our modeling approach is to generate a photo-realistic model, we are interested in reconstructing what the human eye or a camera would observe while moving around in the city. As such, we discuss the discrepancies between these two different sensing modalities in this section.

a) Discrepancies due to different resolution

The beam divergence of the laser scanner is about 15 milliradians (mrad) and the spacing, hence the angular resolution, is about 17 mrad. As such, this is much lower than the resolution of the camera image with about 2.1 mrad in the center and 1.4 mrad at the image borders. Therefore, small or thin objects, such as cables, fences, street signs, light posts and tree branches, are clearly visible in the camera image, but only partially captured in the scan. Hence they appear as "floating" vertices, as seen in the depth image in Figure 6-8.

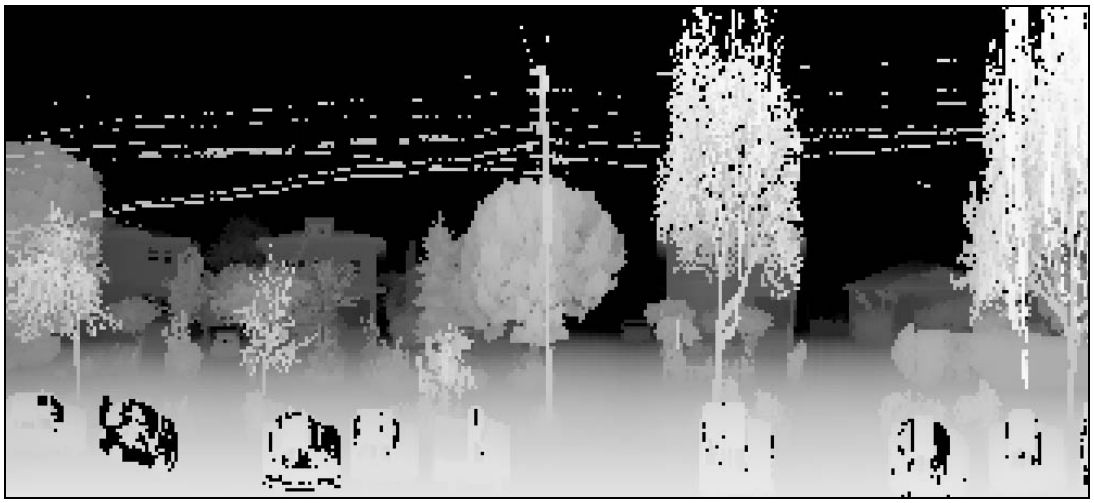


Figure 6-8: "Floating" vertices.

b) Discrepancies due to the measurement physics

Camera and eye are passive sensors, capturing light from an external source, in contrast to a laser scanner, which is an active sensor and uses light that it emits itself. This results in substantial differences in measurements on reflecting and semitransparent surfaces, which are in form of windows and glass fronts frequently present in urban environments.

Typically, there is at least 4% of the light reflected at a single glass/air transition, hence totaling to at least 8 % per window. If the window has a reflective coating, this percentage is even larger. The camera typically sees a reflection of the sky or a nearby building on the window, often distorted or merged with objects behind the glass. Although most image processing algorithms would fail in this situation, the human brain is quite capable of identifying windows. In contrast, depending on the window reflectance, the laser beam is either entirely reflected, most times in a different direction from the laser itself and hence not resulting in any distance value, or is transmitted through the glass. If it hits in the latter case a lambertian surface behind the window, the backscattered light travels again through the glass, as shown in Figure 6-9. The resulting surface reflections on the glass only weaken the laser beam intensity, eventually below the detection limit, but do not otherwise necessarily affect the distance measurement. Thus, the window is quasi non-existent to the laser, and the measurement point is generally not on the window surface, unless the surface is by chance orthogonal to the beam. In case of multi-reflections, the situation becomes even worse as the measured distance is almost random.

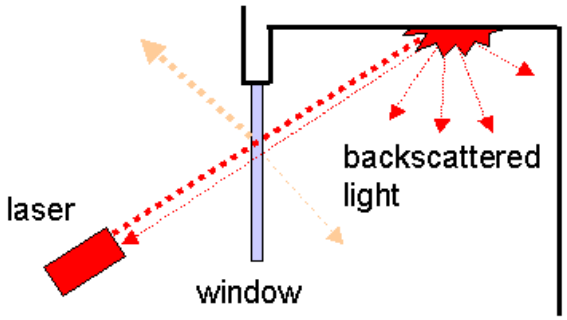


Figure 6-9: Laser measurement in case of a glass window

c) Discrepancies due to different scan and viewpoints

Laser and camera are both limited in that they can only detect the first visible/backscattering object along a measurement direction and can as such not deal with occlusions. If there is an object in the foreground, such as a tree in front of a building, the laser cannot capture what is behind it; hence, generating a mesh from the obtained scan points results in a hole in the building. We refer to this type of mesh hole as *occlusion hole*. As the laser scan points resemble a cylindrical projection, but rendering is parallel or perspective, it is in presence of occlusions virtually impossible to reconstruct the original view without any hole, even for the viewpoints from which data was acquired. An interesting fact is that the wide-angle camera images captured simultaneously with the scans often contain parts of the background invisible to the laser. These images could potentially be used to either fill in geometry based on stereo techniques, or to at least verify the validity of geometry filled in by the interpolation techniques described in the next sections.

In order to make our facade models photo-realistic, we need to devise techniques for detecting discrepancies between the two sensing modalities, removing invalid scan points, and filling in holes resulting from both occlusions and mirroring surfaces; we will describe our approach to these problems in the following sections.

6.4 Multi-Layer Representation

To achieve that the facade model looks reasonable from every viewpoint, it is necessary to complete the geometry for the building facades. As our facades are not only manifolds, but also resemble a height field, it is possible to introduce a representation based of multiple depth layers for the street scenery, similar to the one proposed in [Chang and Zakhor, 1999]. Each depth layer is a scan grid, and the scan points of the original grid are assigned to exactly one of the layers. If there is a point at a certain grid location in a foreground layer, this location is empty in all scene layers behind it and needs to be filled in.

Even though the concept can be applied to an arbitrary number of layers, it is for our problem sufficient to generate only two layers, a foreground and a background. To assign a scan point to either one of the two layers we make the following assumptions about our environment: Main structures, i.e. buildings, are usually (a) vertical, and (b) extend over several feet in horizontal dimension. For each vertical scan S_n corresponding to a column in the depth image, we define the main depth as the depth value that occurs most frequently, as shown in Figure 6-10. The scan vertices corresponding to the main depth lie on a vertical line, and the first assumption suggests that this is a either a main structure, such as a building, or perhaps other vertical objects, such as a street light or a tree trunk. With the second assumption, we filter out the latter class of vertical objects. More specifically, our processing steps can be described as follows:

We sort all depth values $s_{n,v}$ for each column n of the depth image into a histogram as shown in Figure 6-10(a) and (b), and detect the peak value and its corresponding depth. Applying this to all scans results in a 2D histogram as shown in Figure 6-11, and an individual main depth value estimate for each scan. According to the second assumption, isolated outliers are removed by applying a median filter on these main depth values across the scans, and a final depth value is assigned to each column n. We define a "split" depth γ_n for each column n, and set it to the first local minimum of the histogram occurring immediately before main depth, i.e. with a depth value smaller than the main depth. Taking the first minimum in the distribution instead of the main value itself has the advantage that points clearly belonging to foreground layers are splits off, whereas overhanging parts of buildings, for which the depth is slightly smaller than the main depth, are kept in the background layer where they logically belong to, as shown in Figure 6-10.

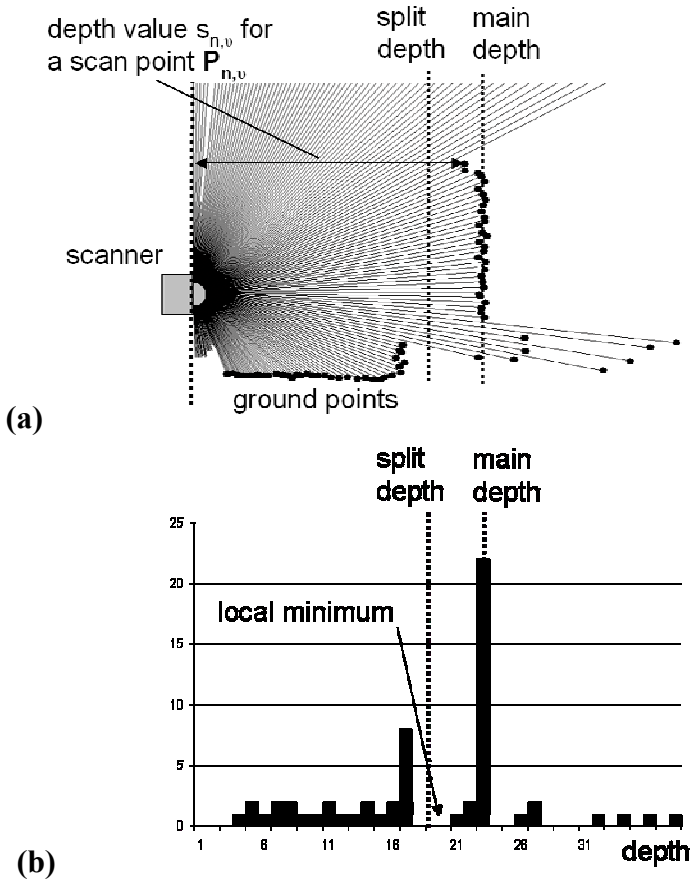


Figure 6-10: Main depth computation for a single scan n; (a) laser scan with rays indicating the laser beams and dots at the end the corresponding scan points; (b) computed depth histogram

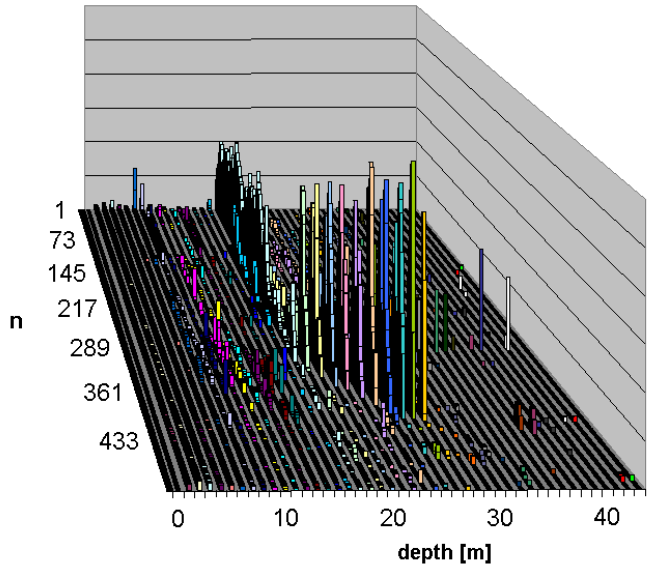


Figure 6-11: Two-dimensional histogram for all scans of a path segment.

A point can be identified as a ground point if its z coordinate has a small value and the direction to its neighbors is approximately horizontal. We prefer to include the ground in our models and hence assign ground points also to the background layer. Therefore, we perform the layer split by assigning a scan point $P_{n,\upsilon}$ to the background layer, if $s_{n,\upsilon} > \gamma_n$ or $P_{n,\upsilon}$ is a ground point, and otherwise to the foreground layer. Figure 6-12 shows an example for the resulting foreground and background layers.

Figure 6-12: Separation into two scene layers; (a) foreground layer; (b) background layer.

Since the steps described in this section assume the presence of vertical buildings, they cannot be expected to work for segments that are dominated by trees; this also applies to the processing steps we introduce in the following sections. As our goal is to reconstruct buildings, path segments can be either omitted or left unprocessed and included "as is" in the city model, if they do not contain any structure. Typical for a tree area is its cluttered geometry, resulting in a large variance among adjacent depth values, or even more characteristically, many significant vector direction changes for the edges between connected mesh vertices. We define a coefficient for the fractal nature of a segment by counting vertices with direction changes greater than a specific angle, e.g. twenty degrees, and dividing them by the total number of vertices. If this coefficient is large, the segment is most likely a tree area and should not be made subject to the processing steps

described in this section. This is for example the case for the segment shown in Figure 6-8.

After layer splitting, all grid locations occupied in the foreground layer are empty in the background layer. The vertical laser does not capture any occluded geometry, and in the next section we will describe an approach for filling these empty grid locations based on neighboring pixels. However, sometimes there is some more data available from other sources, e.g. in our case 3D vertices can be derived from stereo vision and from the horizontal scanner used for navigation. Thus, it is conceivable to use this additional information to fill some holes in the depth layers. Our approach to doing so is as follows:

Given a set of 3D vertices V_i obtained from a different modality, determine the closest scan direction for each vertex and hence the grid location (n,υ) it should be assigned to. As shown in Figure 6-13, each V_i is assigned to the vertical scanning plane S_n with the smallest Euclidean distance, corresponding to column n in the depth image. Using simple trigonometry, the scanning angle under which this vertex appears in the scanning plane and hence the depth image row υ can be computed, as well as the depth $d_{n,\upsilon}$ of the pixel.

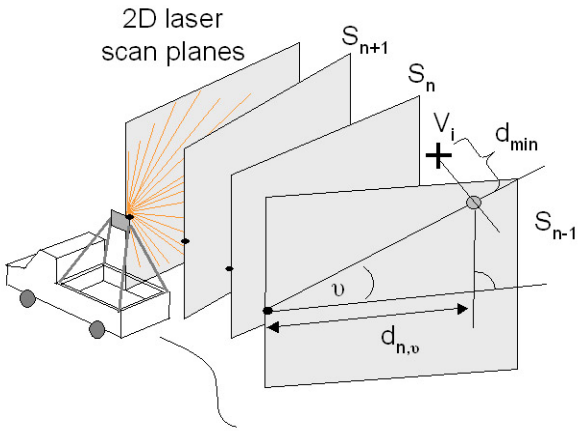


Figure 6-13: Sorting additional points into the layers.

We can now use these additional vertices to fill in the holes. To begin with, all vertices that do not fall onto a background hole location are ignored. If there is exactly one vertex falling onto a grid location, its depth is directly assigned to that grid location; for situations with multiple vertices, the median depth value for this location is chosen. Figure 6-14 compares the background layer before and after sorting in 3D vertices from stereo vision and horizontal laser scans. As seen, some smaller holes can be entirely filled, and the size of others becomes smaller, e.g. the holes in the tall building on the left side caused by the trees. Note that this intermediate step is optional and depends as a matter of course on the availability of additional 3D data.

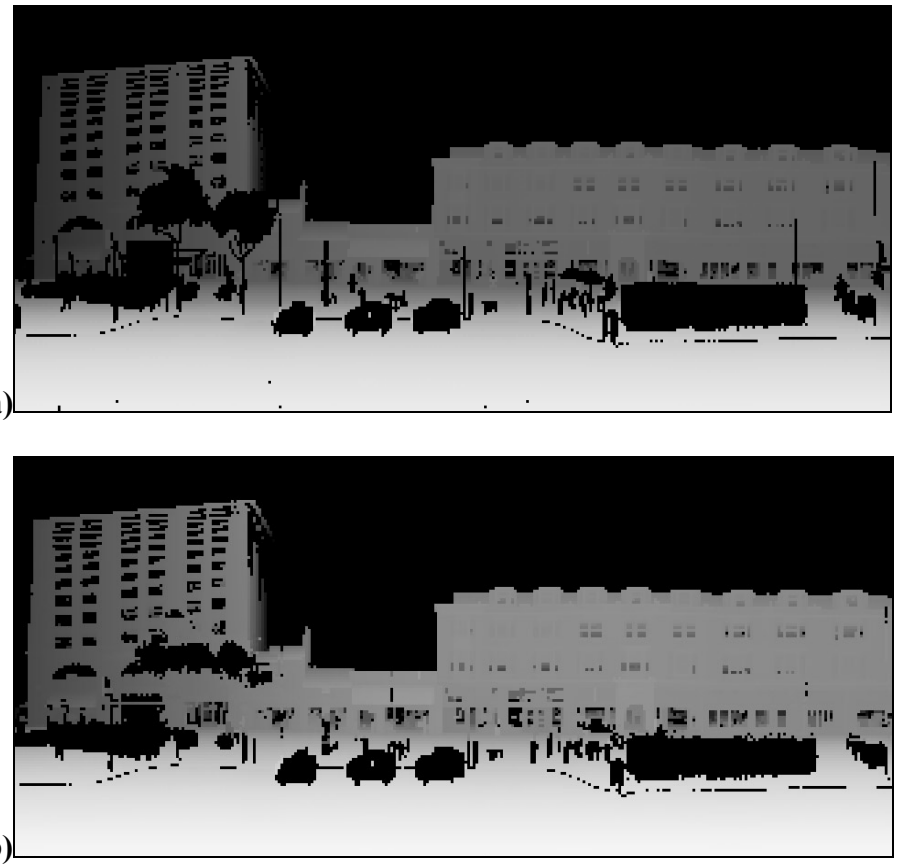


Figure 6-14: Background layer; (a) before and (b) after sorting in some additional points from stereo vision and horizontal laser scans.

6.5 Background Layer Postprocessing and Mesh Generation

In this section, we will describe a strategy to remove erroneous scan points and to fill in holes in the background layer. As stated in the beginning of this chapter, there exists a variety of successful hole filling approaches based on fusing multiple scans taken from different positions. In particular, most previous work on hole filling has been focusing on reverse engineering applications, in which a 3D model of an object is obtained from multiple laser scans taken from different locations and orientations, e.g. with a turn table. Since these existing hole-filling techniques are not applicable to our experimental setup, our approach is to estimate the actual geometry solely based on the surrounding environment and reasonable heuristics. One cannot expect this geometry to be perfect in *all* possible cases, rather to lead to an acceptable result in *most* cases and thus reducing the amount of eventual further manual interventions and postprocessing drastically. Additionally, the estimated geometry could be made subject to further verification steps, such as consistency checks by applying stereo vision techniques to the intensity images captured by the camera.

Our data typically exhibits the following characteristics:

- Occlusion holes, such as those caused by a tree, are large and can extend over substantial parts of a building.
- A significant number of scan points surrounding a hole may be erroneous due to glass surfaces.
- In general, a spline surface filling is unsuitable, as building structures are usually piecewise planar with sharp discontinuities.
- The size of a data set resulting from a city scan is huge, and therefore the processing time per hole should be kept to a minimum.

Based on the above observations, we propose the following steps for data completion:

1. Detecting and removing erroneous scan points in the background layer

We assume that erroneous scan points are due to glass surfaces, i.e. the laser measured either an internal wall/object, or a completely random distance due to multi-reflections. Either way, the depth of the scan points measured through the glass is substantially greater than the depth of the building wall, and hence these points are candidates for removal. Since glass windows are usually framed by the wall, we remove the candidate points only if they are embedded among a number of scan points at main depth. An example of the effect of this step can be seen by comparing the windows of the original image in Figure 6-15(a) with the processed background layer in Figure 6-15(b).

2. Segmenting the occluding foreground layer into objects

In order to determine holes in the background layer caused by occlusion, we segment the occluding foreground layer into objects and project segmentation onto the background layer. This way, holes can be filled in one "object" at a time, rather than all at the same time; we have discovered that more localized hole filling algorithms are more likely to result in visually pleasing models than global ones. We segment the foreground layer by taking a random seed point that does not yet belong to a region and applying a region growing algorithm that iteratively adds neighboring pixels if their depth discontinuity or their local curvature is small enough. This is repeated until all pixels are assigned to a region, and the result is a region map as shown in Figure 6-15(c). For each foreground region, we determine boundary points on the background layer; these are all the valid pixels in the background layer that are close to hole pixels caused by the occluding object.

3. Filling occlusion holes in the background layer for each region

As the foreground objects are located in front of main structures and in most cases stand on the ground, they occlude not only parts of a building, but also parts of the ground. Specifically, a low object such as a car with a large distance to the main structure behind causes an occlusion hole typically in the ground and not in the main structure; this is

because the laser scanner is mounted on top of a rack, and as such has a top down view of the car. As a plane is a good approximation for the ground, we fill in the ground section of an occlusion hole by the ground plane. Therefore, for each depth image column, i.e. each scan, we compute the intersection point between a line through the main depth scan points and a line through ground scan points. The angle υ'_n at which this point appears in the scan marks the virtual boundary between ground part and structure part of the scan; we fill in structure points above and ground points below this boundary in a different way:

Applying a RANSAC algorithm, we find the plane with the maximum consensus, i.e. maximum number of ground boundary points on it, as the optimal ground plane for that local neighborhood. Each hole pixel in column n with $\upsilon < \upsilon'_n$ is then filled in with a depth value according to this plane. It is possible to apply the same technique to the structure hole pixels, i.e. the pixels with $\upsilon > \upsilon'_n$, by finding the optimal plane through the structure boundary points and filling in the hole pixels accordingly. However, we have found that in contrast to the ground, surrounding building pixels do often not lie on a plane. Instead, there are discontinuities due to occluded boundaries and building features such as marquees or lintels, in most cases extending horizontally across the building. Therefore, rather than filling holes with a plane, we fill in structure holes line by line horizontally, in such a way that the depth value at each pixel is the linear interpolation between the closest right and left structure boundary point, if they both exist; otherwise no value is filled in. In a second phase, a similar interpolation is done vertically, using the already filled in points as valid boundary points. This method is not only simple and therefore computationally efficient, it also takes into account the surrounding horizontal features of the building in the interpolation. The resulting background layer is shown in Figure 6-15(d).

4. Postprocessing the background layer

The resulting depth image and the corresponding 3D vertices finally be cleaned up by removing scan points that remain isolated, and by filling small holes surrounded by geometry using linear interpolation between neighboring depth pixels. The final background layer after applying all processing steps is shown in Figure 6-15(e).

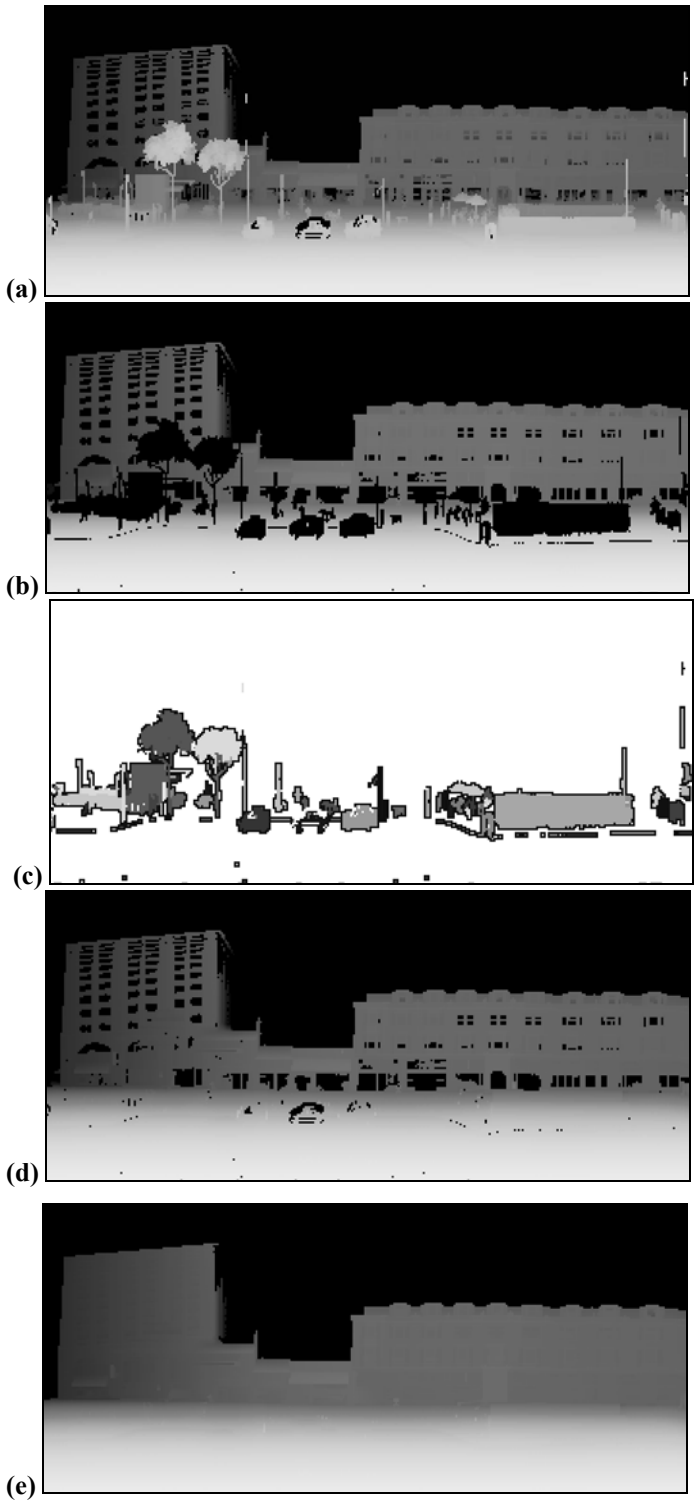


Figure 6-15: Processing steps for a depth image. The individual figures show: (a) initial depth image; (b) background layer after removing invalid scan points; (c) foreground layer segmented; (d) occlusion holes filled, and (e) final background layer after filling remaining holes.

In order to create a mesh, each depth pixel can be transformed back into a 3D vertex, and each vertex $P_{n,\upsilon}$ is connected to a depth image neighbor $P_{n+\Delta n,\upsilon+\Delta\upsilon}$ if

$$| s_{n+\Delta n,\upsilon+\Delta\upsilon} - s_{n,\upsilon} | < s_{\max} \quad \text{or} \quad \cos\varphi > \cos\varphi_{\max}, \tag{6-2}$$

with

$$\cos\varphi = \frac{(\vec{P}_{n-\Delta n,\upsilon-\Delta\upsilon} - \vec{P}_{n,\upsilon}) \cdot (\vec{P}_{n,\upsilon} - \vec{P}_{n+\Delta n,\upsilon+\Delta\upsilon})}{|\vec{P}_{n-\Delta n,\upsilon-\Delta\upsilon} - \vec{P}_{n,\upsilon}| \cdot |\vec{P}_{n,\upsilon} - \vec{P}_{n+\Delta n,\upsilon+\Delta\upsilon}|}. \tag{6-3}$$

Intuitively, neighbors are connected if their depth difference does not exceed a threshold s_{max} or the local angle between neighboring points is smaller than a threshold angle φ_{max}. The second criteria is intended to connect neighboring points that are on a line, even if their depth difference exceeds s_{max}. The resulting quadrilateral mesh is split into triangles, and mesh simplification tools such as Qslim [Garland and Heckbert, 1997] or VTK decimation [Schroeder et al., 1992] can be applied to reduce the number of triangles. Figure 6-16(a) shows an example for a facade mesh obtained directly from triangulating the raw scan points, and Figure 6-16(b) the triangular mesh created after applying the proposed automatic hole filling and foreground removal procedure. It can be seen that the difference between the two meshes is quite substantial, and although the processed mesh may not be as perfect as if edited manually, it appears as an acceptable facade model.

(a)

(b)

Figure 6-16: Generated meshes, (a) original mesh from triangulation of raw scan points; (b) after applying the proposed foreground removal and hole filling procedure

6.6 Automated Texture Mapping

Photo-realism cannot be achieved by using geometry alone; rather, we use images of the actual scene to texture map the geometry and make the facade models appear realistic. As described in Chapter 3, our data acquisition system includes a digital color camera with a wide-angle lens, and its intrinsic parameters and extrinsic parameters in respect to the laser scanners' coordinate system have been calibrated prior to the data acquisition. Since it is synchronized with both scanners and hence taking snapshots at exactly defined times, we can utilize the pose of the horizontal laser scanner and assign a camera pose to each image. This the key for successful photo-realistic model generation in our approach, and a substantial advantage compared to other texture mapping techniques, which attempt to fit laser scans and camera images e.g. by a complicated and error-prone vanishing line detection. After removing the lens distortion in the images, a 3D vertex can be mapped to its corresponding intensity image pixel by a simple projective transformation. Since the 3D mesh triangles are small compared to their distance to the camera, perspective distortions *within* a triangle can be fairly neglected, and each mesh triangle can be texture mapped with the corresponding picture triangle by applying the projective transformation only to the three corner points. We refer to the corresponding triangle in the picture as texture triangle and to the pixel coordinates of its corner points as texture coordinates, in compliance with the common terminology in computer graphics.

As described in Section 6.3, camera and vertical laser scanner have different viewpoints during data acquisition, and in most camera pictures at least some mesh triangles of the background layer are occluded by foreground objects; this is particularly true for triangles that consist of filled-in points, since for them the direct view is certainly occluded. An example of this is given below: Figure 6-17(a) shows an image of a street scenery, and Figure 6-17(b) the triangular mesh back-projected into the image, and as seen, triangles of both the tree and the building project to the same image location, which is only possible since they have been filled in or acquired from different viewpoints than the image. Although the pixel location of the projected background triangles is correct, the corresponding texture is incorrect since it merely corresponds to the occluding foreground objects.

However, after splitting the scan points to the two layers, the foreground geometry is readily identified, and both foreground scan points and triangles can be marked in each camera picture, as shown in Figure 6-17(c) with white color. In order to select specific camera images to texture map a specific mesh triangle in the background layer, we determine whether any of the triangle's corner points projects to a white marked pixel for each picture containing the triangle in its field of view; if this is not the case, the picture is utilizable for texturing the triangle. For the particular wide-angle lens we use, typical building topologies, and typical driving speeds, a background layer point is usually in the field of view of about 10 to 20 pictures. If the data acquisition has taken place on a sunny day, the lighting conditions can for some pictures exceed the dynamic range of the camera, resulting in over-saturated and unusable pictures when facing the sun. Exploiting that in this case there are anomaly few dark pixels, we can identify those images via a color histogram analysis, and prohibit their use for texture mapping, if better images are available. In most situations, we still have a choice among multiple images, and then we choose the most direct view to texture map the triangle. However, if a foreground object and a building facade are too close, some facade triangles may not be visible in any picture and hence cannot be texture mapped at all. For those triangles, no real image data is available and either they are left untextured of an appropriate texture is merely invented. One possible solution is the use of a texture synthesis algorithm as suggested in [Efros and Freeman, 2001], or to create artificial texture for example in a copy-and-paste like fashion.

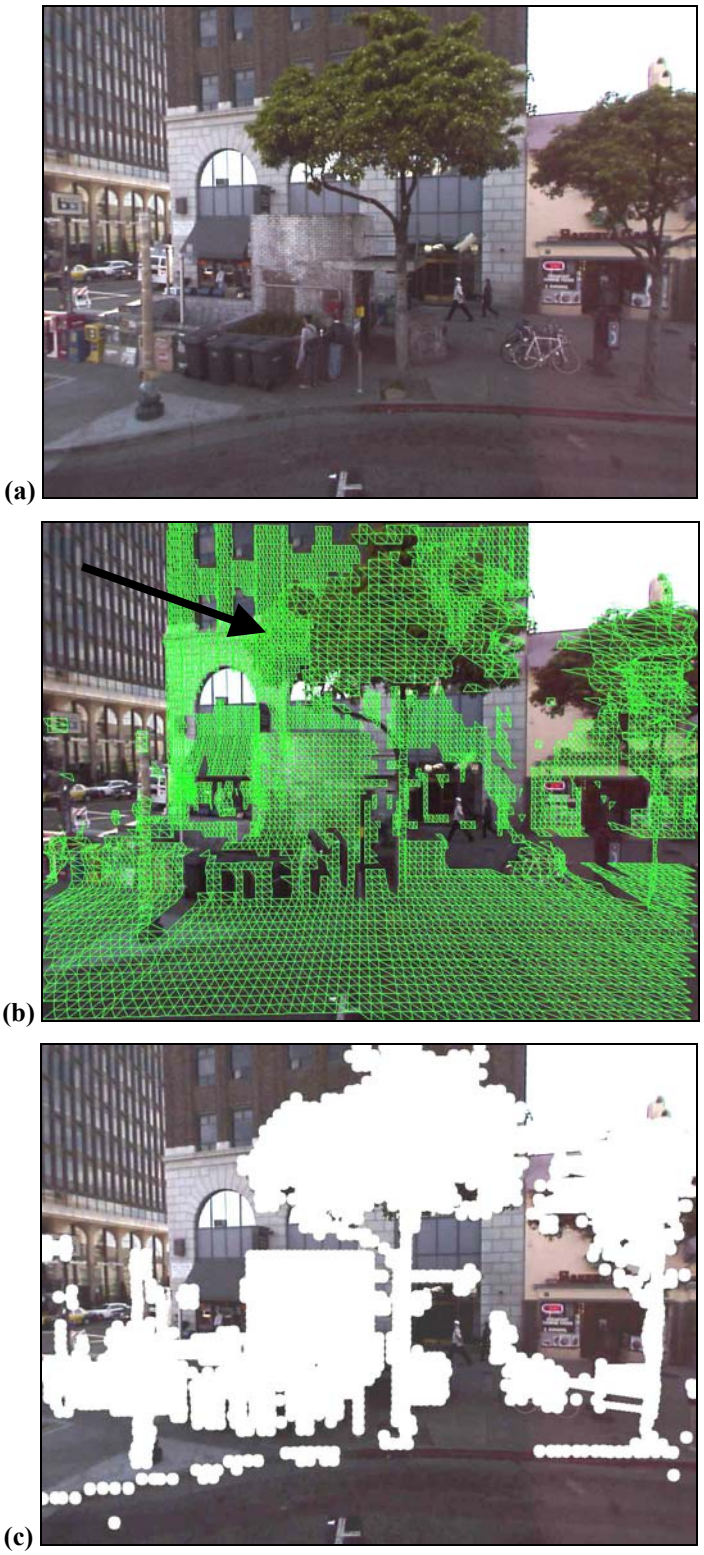


Figure 6-17: Mesh triangles projected into camera images; (a) initial camera image; (b) mesh triangles projected into the image, with some foreground and background triangles projecting to the same image area (arrow); (c) foreground objects marked white.

6.7 Model Optimization for Interactive Rendering

While the actual model generation is finished with the creation of the geometric mesh and the establishment of the corresponding picture area as the texture, the model is in this form virtually unusable for interactive rendering: Displaying an entire city data set would require a graphics card to load and render millions of triangles and thousands of pictures at a time, by far exceeding the capabilities even of modern graphics cards. To overcome this problem, we optimize the original mesh in two additional processing steps: First, we create a texture atlas to reduce the amount of image data necessary for texturing, and second, we create multiple level-of-details and a hierarchical scene graph to manage the amount of triangles and texture.

a) Texture atlas generation

Observing that most parts of a camera image are not utilized since they show foreground objects or are visible in other images at a more direct view, we can reduce the amount of necessary texture imagery drastically by extracting only the parts actually used. The inherent row-column structure of the triangular mesh permits to assemble a new artificial image with a corresponding row-column structure and reserved spaces for each texture triangle. This so-called *texture atlas* is created by copying and warping each individual texture triangle to fit into the corresponding reserved space. Then, the texture coordinates of the mesh triangles are adjusted accordingly, and instead of the numerous original images, the atlas is used to represent the texture. Since in this manner the mesh topology of the triangles is preserved and adjacent triangles align automatically due to the warping process, the resulting texture atlas resembles a mosaic. While the atlas is in fact not precisely metric due to slightly non-uniform spacing between vertical scans, these distortions are small and irrelevant in the context of texture mapping, since they are completely reverted by the graphics card hardware during the rendering process. Figure 6-18 illustrates the atlas generation: From the acquired stream of images, over-saturated frames have been removed. During texture mapping, a corresponding texture triangle in one of the images is assigned to each mesh triangle and then copied into the texture atlas as symbolized by the arrows. In this illustration, only five original images are shown; actually, in this example 58 images of 1024 by 768 pixels size are combined to create a texture atlas of 2559 by 476 pixels. Thus, the texture size is reduced from 45.6 million pixels to 1.2 million pixels, while the resolution remains the same.

Figure 6-18: Automatic texture atlas generation. Texture triangles from various pictures are assembled to one single artificial image. One image has been prohibited for texture mapping due to over-saturation of the camera; for the others, the arrows illustrate the process of copying triangles.

b) Level-of-detail and scene graph generation

An entire city consists of tens of millions triangles and even after atlas generation, the texture for an entire city district can easily total to some hundred megabytes, thus exceeding the rendering capabilities of any existing graphics card. But since for any given viewpoint only a small subset of the entire model has to be shown at the highest resolution, an elegant solution is to store multiple level-of-details (LOD) for each facade mesh and select the appropriate LOD during rendering. To obtain a lower LOD, we resize the atlas to a fraction of its original dimensions via bicubic interpolation and scale the texture coordinates accordingly. For the geometry, the amount of triangles is reduced with the Qslim mesh simplification algorithm [Garland and Heckbert, 1997], which removes and regroups triangles of a given mesh based on edge collapsing. The atlas texture coordinates of the remaining mesh vertices can thereby simply be reused; since the atlas is approximately metric, the new, larger triangles are textured with the composed texture from several original texture triangles. Small distortions due to the warping process in the atlas composition are not noticeable from the far-away view for which the lower LODs are used.

Already one single path segment can result in a quite large facade mesh, and its megabytes of texture can reach dimensions that are not supported by OpenGL anymore. Additionally, the entire segment has to be rendered in the highest LOD, even if the viewer is close to only a small part of it, e.g. requiring several complete facade meshes at highest LOD while the viewer is at a street intersection. Therefore, it is not efficient to

use such a large facade mesh as the smallest LOD unit, rather, we subdivide the facade mesh together with the texture atlas along vertical planes and generate different LODs for each sub-mesh, thus enabling enhanced flexibility for switching LODs. Dividing all LOD meshes along the same vertical planes has the advantage that seams between different LODs are far less noticeable. As illustrated in Figure 6-19, all sub-meshes are then combined in a hierarchical scene graph, controlling the switching of the LODs dependent on the viewer's position and establishing compliance with the limitations of graphics hardware. This enables us to render the enormous amount of both geometry and texture even with a standard web based VRML renderer such as Cosmo player.

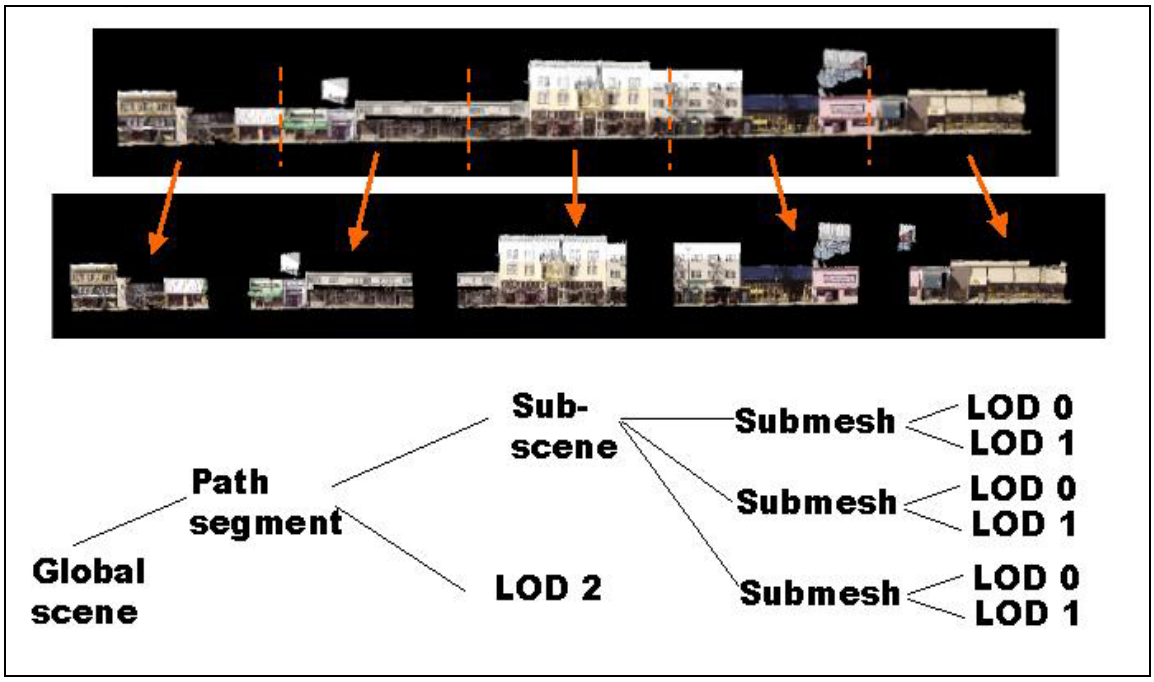


Figure 6-19: Subdividing the mesh of a path segment into sub-meshes and generation of a scene graph

7 Airborne Model Generation and Model Fusion

While the facade models generated in the previous chapter offer a virtual view as seen from street level, they do not contain any information about roofs, terrain, or building structures behind the facades. Essentially, these models are the virtual equivalent to a Hollywood city, and it is apparent that solely facades are not sufficient for virtual fly-thrus. Therefore, we outline in this Chapter methods of enhancing the facades with roofs and terrain shape from airborne laser scans. From these scans, we create a DSM, convert it to a surface mesh, and texture map it with aerial color images. Then, we merge the airborne surface mesh and the facade models, in order to obtain a complete model suitable for both walk- and fly-thru. It turns out that one of the greatest advantages of our particular localization method is its inherent registration of the facade models with a global model, hence substantially facilitating subsequent model merging.

7.1 Resampling and DSM Generation from Airborne Laser Scans

Due to advances in technology, airborne laser scans have become a data source for 3D modeling in recent years. To scan a city from an airborne view, a far-range 2D scanner is mounted on board a plane, so that it scans along a line perpendicular to the flight direction. Due to the plane's forward motion, the scan line sweeps over the ground. The unpredictable roll and tilt motion of the plane generally destroys the inherent row-column order of the scans. Therefore, GPS and INS readouts are captured simultaneously along with the scanning process, and 3D vertices are directly obtained from the position and orientation of the scanner and the measured distance. Thus, the scans may be interpreted as an unstructured set of 3D vertices in space, with the x,y coordinates specifying the geographical location and the z coordinate the altitude. Both flight and data conversion is usually done by a professional company, and as a customer one simply buys the resulting set of 3D scan points.

Typically, the accuracy of one single scan point is in the range of one foot, whereas the non-uniform density of scan points is a function of the flight altitude and can be specified in points per square meter. In order to further process the scan efficiently, it is advantageous to resample the scan points to a DSM, which is a regular row-column array over the geographical area, with a z value assigned to each grid cell. Unfortunately, this step generally reduces the lateral resolution to the grid spacing, and since scan sweeps overlap and single measurements are taken under different oblique angles, there are potentially different z values for identical (x,y)-locations, e.g. multiple scan points on a vertical wall or on the roof above. Furthermore, near discontinuities, it is random where exactly the laser hits the surface; thus, without further assumptions, the location of an edge cannot be determined more accurately than the sample density. This causes edges to be jittery in a DSM obtained from laser scans, and the accuracy for determining orientation of edges is not as high as with aerial photos. However, the enormous advantage of airborne laser scans is that they provide directly correct 3D coordinates for

the geometry, and no error-prone camera parameter estimation, line or feature detection and matching has to be performed.

To transfer the scans to a DSM, we define a row-column grid in the ground plane and sort scan points into the grid cells. The density of scan points varies and hence there are cells with no scan point and others with multiple scan points. Since both the percentage of empty cells and the resolution of the DSM depend on the grid spacing, a compromise must be found, leaving few cells without a sample while maintaining the resolution at an acceptable level. In our case, we have chosen to select a square cell size of 0.5 by 0.5 meter, which fills about half of the cells. We create the DSM by assigning each cell the highest z value occurring among its member points, so that overhanging rooftops of buildings are preserved while lower points on sidewalls are suppressed. The empty cells are filled by nearest-neighbor-interpolation, in order to preserve sharp edges. Each grid cell can be interpreted as a vertex with x- and y- coordinates the location of the cell center and the z coordinate the altitude value, or as a pixel at (x,y) with a gray intensity proportional to z. Figure 7-1 shows an example of a point cloud and the resulting DSM encoded as a gray image.

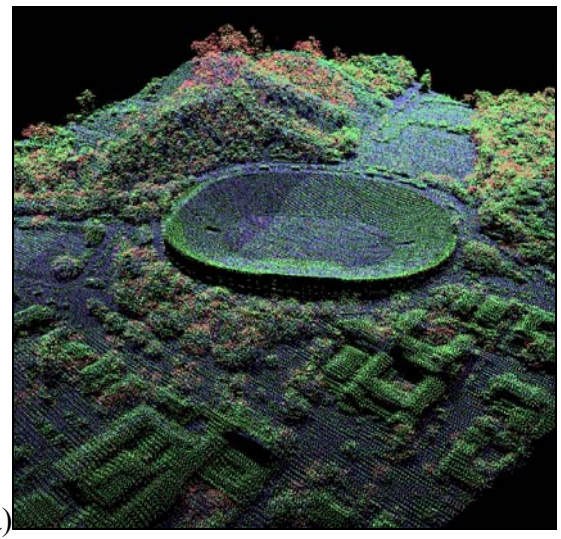
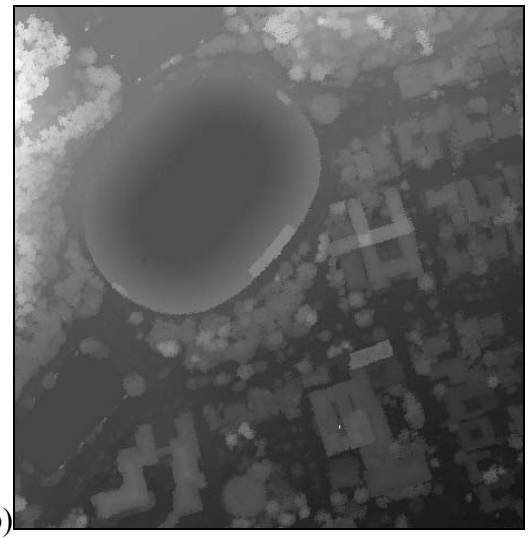
(a)  (b) 

Figure 7-1: (a) Raw scan points and (b) resampled DSM as gray image

Note that a DSM is one of the most basic representations for a model. In fact, any polygonal model in which buildings are represented in a “shoe-box” fashion can easily be transformed into a DSM; similarly, a DSM can be obtained from stereo vision or SAR. Hence, the mesh generation and model fusion steps described in the following sections are by no means limited to the case of airborne laser scans; rather, they comprise a very general approach to complement facade models with data from airborne view.

7.2 Airborne Model from the DSM

Previous work has focused on the extraction of polygonal building models from DSMs, often supported by additional data. Similar to airborne stereo vision, most approaches attempt to match certain primitive polygonal building types to the DSM, either from a set of predefined types, or more sophisticated, derived from planar surfaces in the DSM itself and eventually additional data. Example for model reconstruction from laser scans only can be found in [Weidner and Förstner, 1995], [Vosselman, 1999] and [Maas, 2001]. Other approaches avoid the problem of jittery edges in a DSM by utilizing multiple additional data sources such as digital ground plans ([Haala and Brenner, 1997], [Brenner et al., 2001], [Vosselman and Dijkman, 2001]). It has also been popular to extract edge locations and directions from aerial photos ([Förstner, 1999], [Ameri and Fritsch, 2000]) or hyperspectral images ([Haala and Brenner, 1999]) at higher resolution, and to combine this accurate edge information with the accurate altitude information from the DSM. While the advantage of these model-based approaches is their robust reconstruction of geometry, even in the presence of erroneous scan points and low sample density, they are often highly dependent on the shape assumptions that are made. In particular, the results are poor if many non-conventional buildings are present or if buildings are surrounded by trees, conditions that are particularly true of the Berkeley campus. Although the resulting model may appear "clean" and precise, the geometry and location of the reconstructed buildings is not necessarily correct if the underlying shape assumptions are invalid.

From the ground-based acquisition described in the previous section an accurate model of the building facades is readily available, and as such, we are primarily interested in adding the complementary roof and terrain geometry. Hence, we can apply a different strategy to create a model from airborne view, namely transforming the cleaned-up DSM directly into a triangular mesh and reducing the number of triangles by simplification. The advantage of this method is that the mesh generation process can be controlled on a per-pixel level; we exploit this property in the model fusion procedure described in Section 7.3. Additionally, this method has a low processing complexity and is robust: Since no pre-defined models are required, it can be applied to buildings with unknown shapes, even in presence of trees. Admittedly, this comes at the expense of a larger number of polygons.

7.2.1 Processing the DSM

The DSM contains not only the plain rooftops and terrain shape, but also many other objects such as cars, trees, etc. Roofs, in particular, look "bumpy" due to a large number of smaller objects such as ventilation ducts, antennas, and railings, which are impossible to reconstruct properly at the DSM's resolution. Furthermore, scan points below overhanging roofs cause ambiguous altitude values, resulting in jittery edges. In order to obtain a more visually pleasing reconstruction of the roofs, we apply several processing steps:

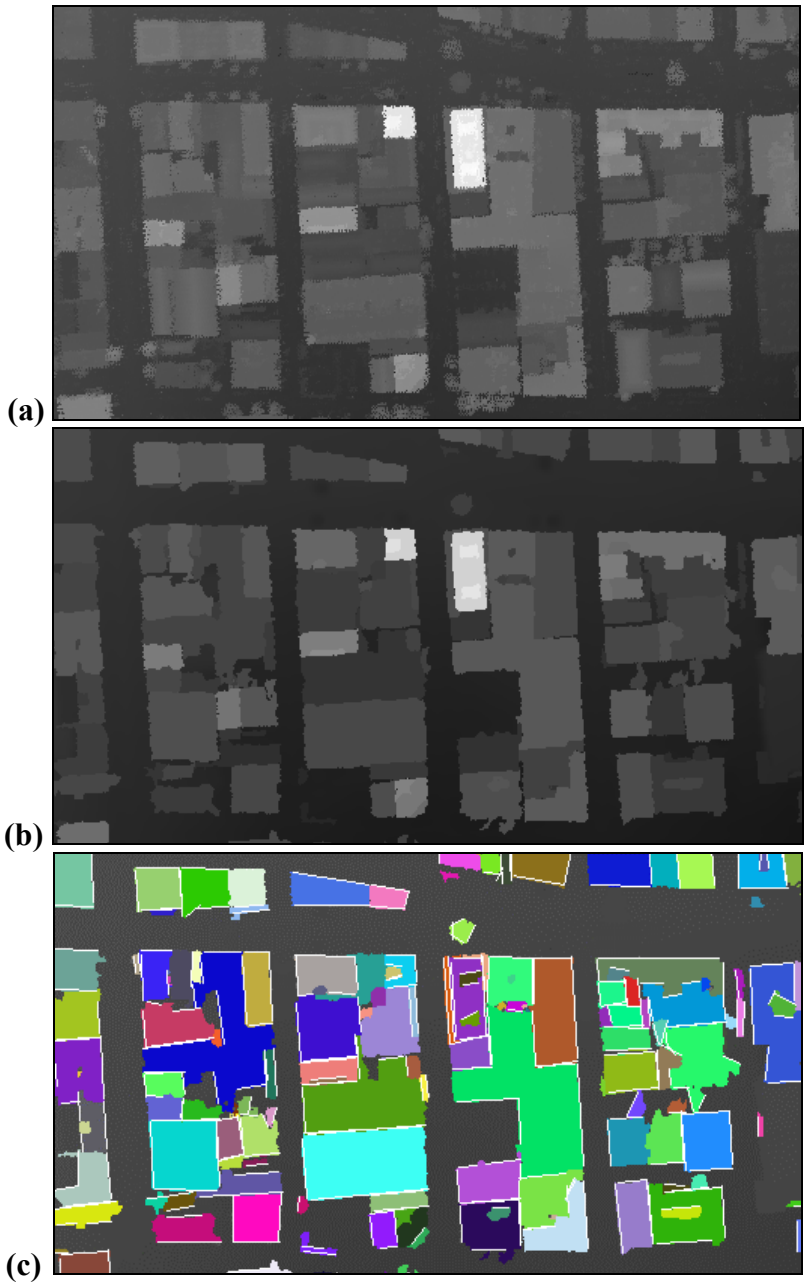


Figure 7-2: Processing steps for DSM; (a) DSM obtained from scan point resampling; (b) DSM after flattening roofs; (c) segments with RANSAC lines in white.

The first step is aimed at flattening "bumpy" rooftops. To do this, we first apply to all non-ground pixels a region-growing segmentation algorithm based on depth discontinuity between adjacent pixels. Small, isolated regions are replaced with ground level altitude, in order to remove objects such as cars or trees in the DSM. Larger regions are further subdivided into planar sub-regions by means of planar segmentation. Then, small regions and sub-regions are united with larger neighbors by setting their z values to the larger region's corresponding plane. This procedure is able to remove undesired small objects

from the roofs and prevents rooftops from being separated into too many cluttered regions. The resulting processed DSM for Figure 7-2(a) is shown in Figure 7-2(b).

The second processing step is intended to straighten jittery edges. We re-segment the DSM into regions, detect the boundary points of each region, and use RANSAC [5] to find line segments that approximate the regions. For the consensus computation, we also consider boundary points of surrounding regions, in order to detect even short linear sides of regions, and to align them consistently with surrounding buildings; furthermore, we reward additional bonus consensus if a detected line is parallel or perpendicular to the most dominant line of a region. For each region, we obtain a set of boundary line segments representing the most important edges, which are then smoothed out. For all other boundary parts, where a proper line approximation has not been found, the original DSM is left unchanged. Figure 7-2(c) shows the regions resulting from processing Figure 7-2(b), superimposed with the corresponding RANSAC lines drawn in white. Compared with Figure 7-2(b), most edges are straightened out.

7.2.2 Textured Mesh Generation

Since the DSM has a regular topology, it can be directly transformed into a structured mesh by connecting each vertex with its neighboring ones. The DSM for a city is large, and the resulting mesh has two triangles per cell, yielding 8 million triangles per square kilometer for the 0.5 m × 0.5 m grid size we have chosen. Since many vertices are coplanar or have low curvature, the number of triangles can be drastically reduced without significant loss of quality. We use the Qslim mesh simplification algorithm [Garland and Heckbert, 1997] to reduce the number of triangles. Empirically, we have found that it is possible to reduce the initial surface mesh to about 100,000 triangles per square kilometer at highest level-of-detail without noticeable loss in quality.

Using aerial images taken with an uncalibrated camera from unknown poses, we texture-map the reduced mesh in a semi-automatic way: A few correspondence points are manually selected in both the aerial photo and the DSM, taking a few minutes per image. Then, both internal and external camera parameters are automatically computed and the mesh is texture-mapped. Specifically, a location in the DSM corresponds to a 3D vertex in space, and can be projected into an aerial image if the camera parameters are known. We utilize an adaptation [Araujo et al., 1998] of Lowe's algorithm [Lowe, 1991] to compute the optimal camera pose by minimizing the difference between selected correspondence points and computed projection. After the camera parameters are determined, for each geometry triangle, we identify the corresponding texture triangle in an image by projecting the corner vertices. Then, for each mesh triangle the best image for texture-mapping is automatically selected by taking into account resolution, normal vector orientation, and occlusions. Figure 7-3 shows the resulting texture-mapped airborne model.

Figure 7-3: Texture-mapped airborne model.

7.3 Merging Ground-Based Models and Airborne Surface Mesh

In the previous sections, we have described the creation of a DSM and a textured surface mesh, and in this section we describe an approach to merge the ground-based facade models with this surface mesh. Since the global localization methods of Chapter 5 correct the initial path according to the edges in the DSM, one of the core problems of model merging, namely the model registration, is already automatically solved. Hence, the remaining difficulty is to combine facades and surface mesh to a single, consistent model.

Common approaches for fusing meshes, such as sweeping and intersecting contained volume [Stamos and Allen, 2002], or mesh zippering [Turk and Levoy, 1994], require a substantial overlap between the two meshes. This is not the case in our application, since both views are almost perfectly complementary. Additionally, the two meshes have entirely different resolutions: with about 10 to 15 cm, the resolution for the facade models is almost an entire order of magnitude higher than for the airborne surface mesh. Furthermore, it has to be guaranteed that parts of the model fit together even if they are displayed at different level-of-details, which is inevitable to enable interactive rendering. Rather than creating a perfectly consistent CAD model, our goal is a photo-realistic virtual exploration of the city, and hence satisfying visual appearance is more important than CAD model properties such as watertightness. Both meshes are generated automatically, and given the complexity of a city environment, it is inevitable that some parts are only partially captured or erroneous, potentially resulting in substantial discrepancies between the two meshes.

Due to the higher resolution, it is reasonable to give preference to the ground-based facades wherever available, and use only roof and terrain shape from the airborne mesh. Instead of searching through the airborne mesh and removing triangles for which ground-based geometry is available, it is more efficient to consider this redundancy already in the mesh generation step. For all vertices of the ground-based facade models, we mark the corresponding cells in the DSM; furthermore, we identify and mark all areas classified as foreground by our automated facade processing. These marks control the subsequent airborne mesh generation, specifically, during the generation of the airborne mesh, vertices at facade positions are not connected, and the z values for the foreground areas are replaced by the ground level estimate. The latter step is necessary to enforce consistency, since it removes foreground objects, which have been deleted during the facade model generation, also in the airborne mesh. Figure 7-4(a) shows the DSM with facade areas marked in black and foreground areas marked in white, and Figure 7-4(b) shows the resulting airborne surface mesh with the corresponding facades removed and the foreground areas leveled to DTM altitude.

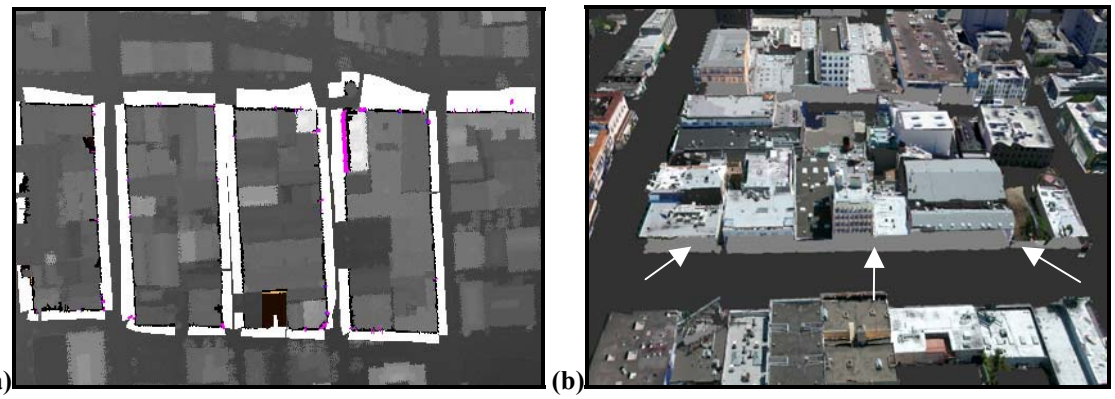


Figure 7-4: Removing triangles from the airborne surface mesh where ground-based facades are available; (a) foreground (white) and facades (black) marked in the DSM; (b) resulting mesh with corresponding facades triangles removed (white arrows).

Now the facade models can be put in place, but facades and airborne mesh do not match perfectly due to their capturing viewpoint and different resolution: Any ground-based vertex set back from the facades, e.g. in a house entrance, causes the corresponding cell to be omitted during meshing. As a result, the removed geometry is slightly larger than the actual ground–based facade to be placed in the corresponding location. To solve this discrepancy and make the mesh transitions less noticeable, we fill the gap with additional triangles joining the two meshes, and we refer to this step as “blending”. Our approach to creating such a blend mesh is to first create a mesh overlap artificially by extruding the buildings along an axis perpendicular to the facades, and then shift the location of the “loose end” vertices to the closest airborne mesh surface. The outline of this procedure is illustrated in Figure 7-5; it is similar to the way plumb is used to close gaps between

windows and roof tiles. These blend triangles, and the non-textured triangles that were out of the camera's field of view, are finally texture mapped with the aerial imagery, and as such they attach at one end to the ground-based model, while fitting at the other end to the airborne model. Figure 7-6 shows the blending steps for a concrete example.

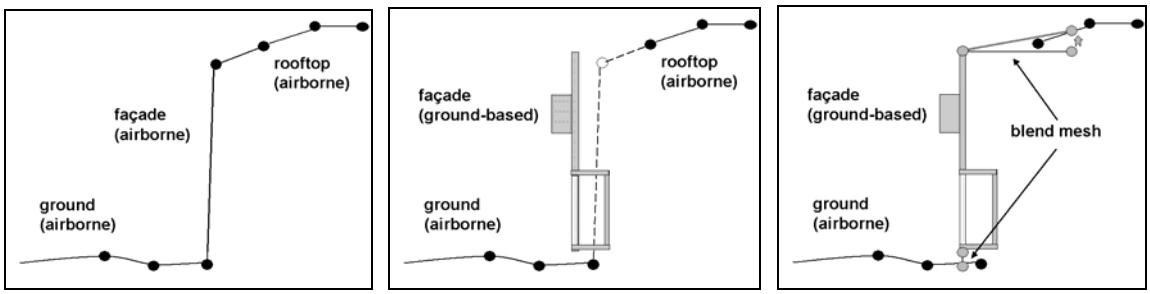


Figure 7-5: Steps to create blend triangles. Shown is a vertical cut through a facade mesh; (a) initial airborne model; (b) triangles of airborne model removed and ground-based model placed in the resulting gap; (c) blending both meshes with extruded triangles.

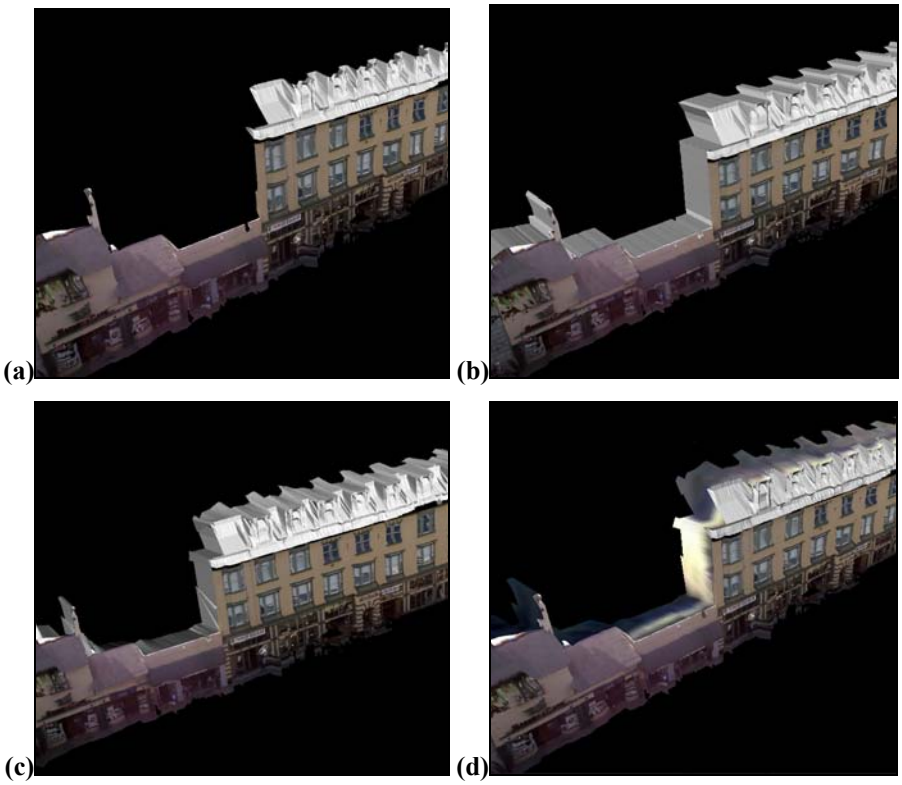


Figure 7-6: Creation of a blend mesh; (a) initial facade model; (b) facades extruded; (c) "loose ends" adjusted to airborne mesh surface; (d) blend triangles texture mapped

8 Results

In the preceding chapters, we have proposed a new approach to acquire data from ground-based view and to create photo-realistic building facade models in a fast and automated way. This chapter is devoted to applying the described algorithms to real-world data and to analyzing both the efficiency of our algorithms and the quality of the obtained results.

8.1 Ground-Based Data Acquisition

The ground-based data was acquired during a 37-minute-drive in Berkeley, California, for which the speed was only limited by the normal traffic conditions during business hours. Starting from Warring Street in the Berkeley Hills, we descended through residential areas, passed through Telegraph Avenue, then further down along Bancroft Way and around the U.C. Berkeley campus. Finally, we went in clockwise loops around the blocks between Shattuck Avenue and Milvia Street, while always driving two blocks southwards on Shattuck and only one block northwards on Milvia. As our devices are mounted only on the right side of the truck, driving in loops is the only way to acquire data for both sides of the streets, and hence obtain the facades completely for an entire area. The driven path had a total length of 10.2 kilometers, and while driving, we captured 148,665 vertical and horizontal scans, consisting of 39.74 million 3D scan points along with a total of 7,200 images.

Roughly the first third of the driven path, before we reach Telegraph Avenue, is on a hillside and contains steep slopes. It leads through residential and campus areas for which often only trees are in the field of view of our sensors, and it also passes a huge empty parking lot and a plaza. This part of the path was intended to test the robustness of the MCL localization on edge maps from the DSM, while a 3D facade reconstruction is not possible, due to occlusion or complete absence of building structures. To analyze our geometry reconstruction techniques, we use the 6769 meters long path segment where urban structures are present, starting from Telegraph Avenue to the end of the path. During this 24-minute part of the path, we captured 107,082 vertical scans, consisting of a total of 28.63 million scan points. For the last 11 minutes or 3043 meters of our path, we drove in loops around the downtown blocks between Shattuck Avenue and Milvia Street, and we captured both laser scans and camera images. Thus, for this area, we have all measurement modalities completely available, and hence we can both create texture mapped facade models and merge these with the airborne surface mesh from the DSM.

8.2 Tracking

We apply the scan-to-scan matching and the path computation described in Chapter 4 to the acquired horizontal laser scans and obtain a series of relative 3-DOF pose estimates. We have analyzed the accuracy of the scan matching process already in Chapter 4.2 and

determined the approximate accuracy of the matching result to be better than 0.03 degrees for the rotation and better than 1 cm for the translation parameters for areas providing a "normal" amount of features.

To evaluate the effectiveness of the adaptive subsampling of horizontal scans for the path computation, we compare the resulting path for a downtown Berkeley city block, acquired in 118 seconds driving time. Figure 8-1 shows path and scan points obtained with a fixed subsampling factor of 10, and Figure 8-2 using the adaptive subsampling method, respectively. In both figures, the computed path is represented by the black line going around the block; orthogonal to it, the scanning direction of the vertical laser scanner for each computed position is shown. Also shown are the points of the horizontal laser scan for each position, superimposing to a footprint of the building facades. The alignment of these scan points is a measure of the computed path accuracy; if the path is correct, the points of a building wall measured from different positions should ideally lay on a sharp line, otherwise they will form rather a blurred strip.

For the fixed subsampling factor used in Figure 8-1, the areas where the truck moved slowly or stopped completely can be clearly identified by looking at the drawn vertical scanner directions: positions are computed at fixed time intervals, and therefore the density of estimates is higher during slow motion. For these parts of the path, the scan point alignment is visibly worse than for the rest, as shown in the detailed view, because the accumulation of the estimation noise leads to position inaccuracy. Figure 8-2 shows the same path calculated with adaptive subsampling, given a desired minimum of 80 centimeters and a maximum of 150 centimeters per step. As seen in the detailed view, the scan alignment is significantly better, regardless of speed or stopping times during data acquisition. This also results in correct angles between the roads. Note that the upper and lower part of the traveled path are not parallel, because we changed lanes during driving; this lane change occurs in the path as evidenced by the decrease in distance between computed path and building facades from left to right in the upper trajectory of Figure 8-2. Furthermore, the angle between the two roads and Shattuck Av. on the right side of the Figure 8-2 is actually not 90^0, also computed correctly by our algorithm.

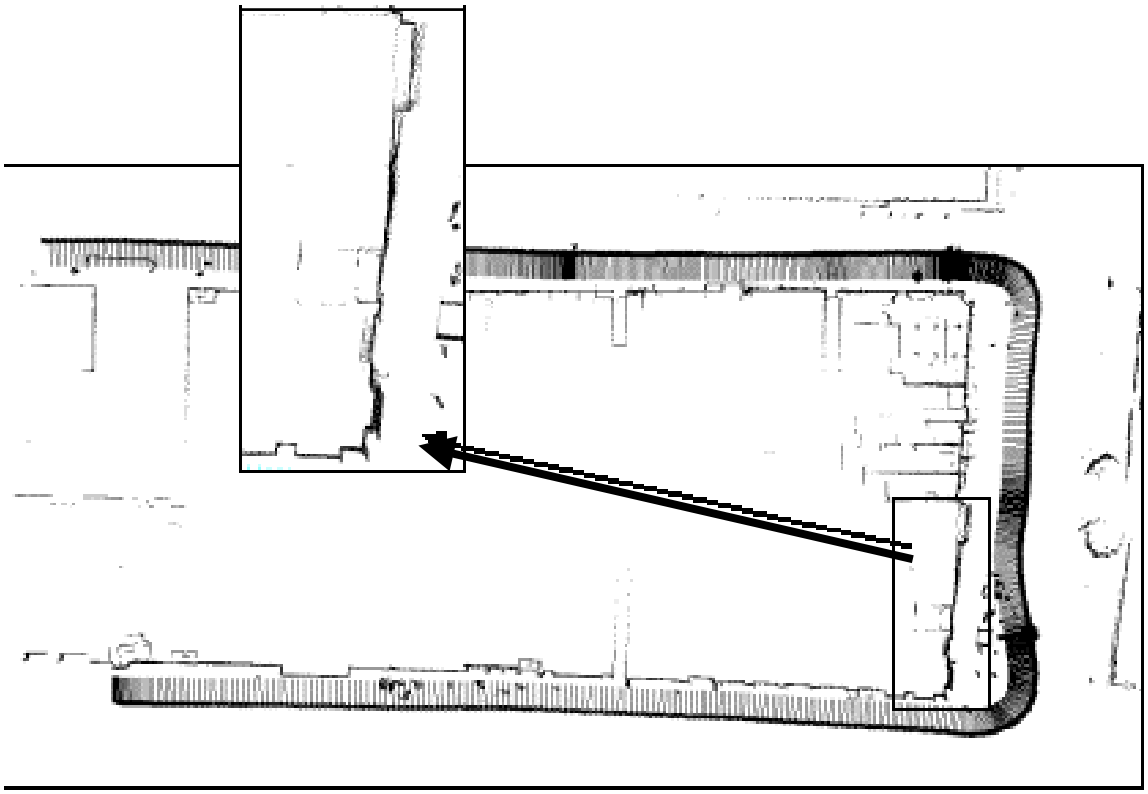


Figure 8-1: Path computed with fixed subsampling factor of 10

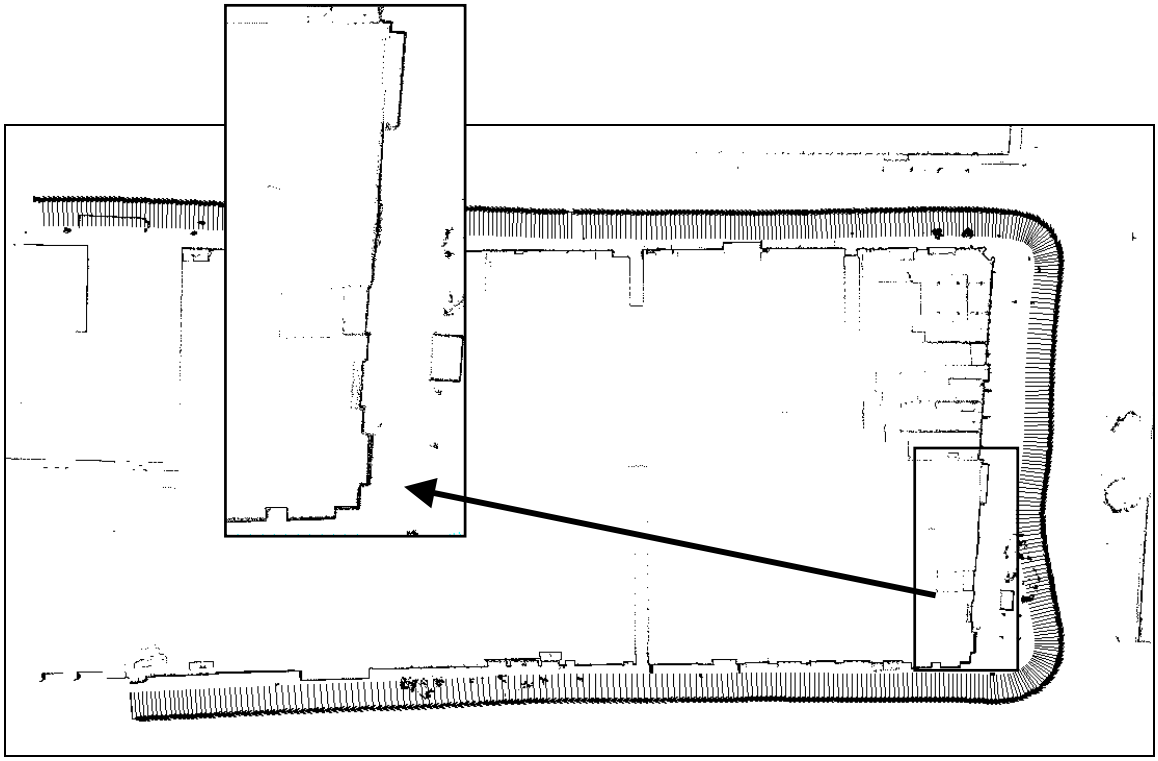


Figure 8-2: Path computed using adaptive subsampling

We have applied the scan matching and initial path computation to the entire driven path. With a minimal displacement of 80 cm and a maximum displacement of 150 cm for a single step, the adaptive subsampling yields 10109 relative estimates for the entire 10.2 km path. For obtaining this 10109 relative steps, 6753 additional intermediate scan matches were computed in the adaptive subsampling and not regarded as a full step, since the distance to the end position of the previous step was too small, so that we had in fact to perform a total of 16862 scan matches.

Figure 8-3: Path computed by concatenating relative pose estimates obtained in the scan-to-scan matching process, superimposed on top of the DSM

Figure 8-3 shows the path computed by concatenating these relative steps, superimposed on top of the DSM. The sequence of turns and straight drives is clearly recognizable, and the error between relative poses in the path is small within a certain neighborhood. However, it is apparent that the global position becomes increasingly incorrect for longer driving, and for area traversed multiple times the individual facades would not match consistently.

8.3 Global Localization Based on Aerial Images

In order to correct the apparent errors in global pose with the methods described in Chapter 5, we first compute an edge map from an aerial photo. As a manual step, we either mark the approximate starting point of our data acquisition in the aerial photo, or enter the name of the corresponding road intersection. If desired, this step could easily be automated as well if a low-cost, low-accuracy GPS is used to obtain the starting position. However, due to the small effort that it takes, we do not consider this step worth to be automated, especially since it has to be done only once per data acquisition, independently from the length of the subsequent driving.

8.3.1 Edge Map Computation

While perspective-corrected black and white photos with a 1-meter resolution, registered to digital roadmaps, are readily available from the United States Geological Survey (USGS), we choose to use a higher contrast aerial photograph obtained from Vexcel Corporation, CO, USA, witch has a resolution of one foot per pixel and covers the entire area of our data acquisition. This aerial photo is not a perfect ortho-photo, however, it has been taken approximately perpendicular to the ground. As shown in Figure 8-4, we have analyzed the metric properties of the aerial image by superimposing a metrically correct digital roadmap available from the USGS web site. Although the area marked by the dashed rectangle is not completely flat, we could verify that for this area the effect of perspective distortion within the ground plane is negligible; it includes the entire urban path segment starting from Telegraph Avenue. Hence, the photo can be used as a metric global reference for this area, and applying a Sobel filter, we compute the edge map shown in Figure 8-5, with a detailed view on Shattuck Avenue. However, the superposition also revealed the photo's metric to be erroneous for the hillside area traversed in the first part of the path, due to significant perspective shifts of the roads caused by the elevation in the Berkeley Hills. Thus, it cannot be used for this area.

Figure 8-4: Aerial image superimposed with digital roadmap (white)

Figure 8-5: Edge map derived from aerial photo

8.3.2 Localization by Maximizing Congruence

We have applied the global pose correction based on maximizing congruence to the entire path segment starting from Telegraph Avenue, for which the aerial image can be regarded as metric. As already described in Section 5.4.1, we have used the digital roadmap to coarsely adjust the initial path obtained from scan-to-scan matching. Then, as shown in Figure 8-6, we have applied the congruence maximization technique from Section 5.4.2 to further refine the global pose. Figure 8-6(a) shows the roadmap-adjusted path, and Figure 8-6(b) and (c) the corrected path superimposed on the edge map and the aerial photo, respectively. Notice that the roadmap-adjusted path shown in Figure 8-6(a) is situated outside the edge boundaries of the actual road, and is visibly incorrect. Furthermore, the corresponding laser scans do not match the building edges, confirming global pose inaccuracy in Figure 8-6(a). As seen, these problems are clearly absent in Figure 8-6(b). Due to the averaging over multiple correction vectors and relying on features such as trees, the correct location in respect to the ground can be found even in presence of a perspective rooftop shift of the two taller buildings in the middle of the image. Notice that we also correctly recover the lane change from the right to the left lane after the crossing in the middle (the shown Telegraph Avenue is a two-lane one-way street).

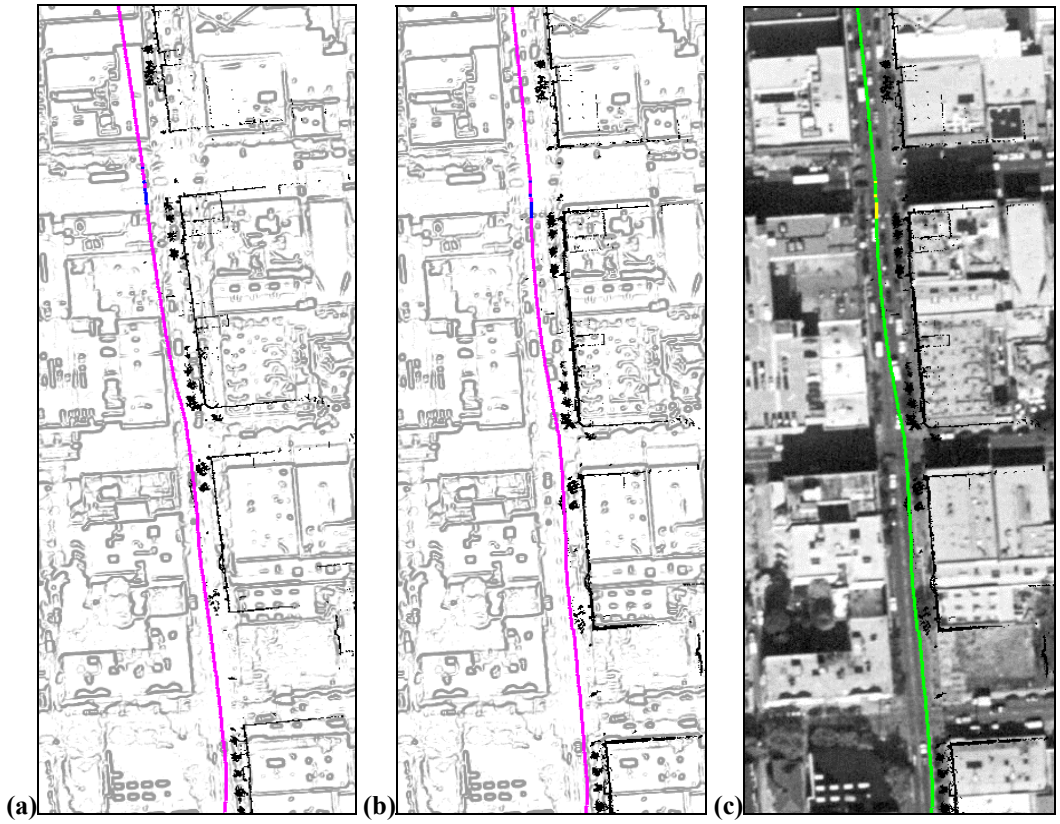


Figure 8-6: Global pose by maximizing congruence. The figures show path and laser scans superimposed on edge images for (a) original path; (b) path corrected by maximizing congruence; (c) corrected path superimposed over original aerial image

While we have found that the correction works well in downtown areas as shown above, it is unfortunately less reliable in suburban areas with houses hidden among trees. In particular for residential areas, the selection of heuristic parameters such as search window and correction vector weight turns out to be crucial for the success of the method. While we have been able to recover the entire path for one specific parameter set, we have always lost track of the vehicle for others, similar sets while looping in a particular residential area. Hence, we believe that this method cannot be reliably applied to recover trajectories through arbitrary residential areas. This method is also incapable of pursuing multiple distinct pose hypotheses in ambiguous situations, and no measure for the pose uncertainty is incorporated, since pose is represented only as one discreet set of parameters.

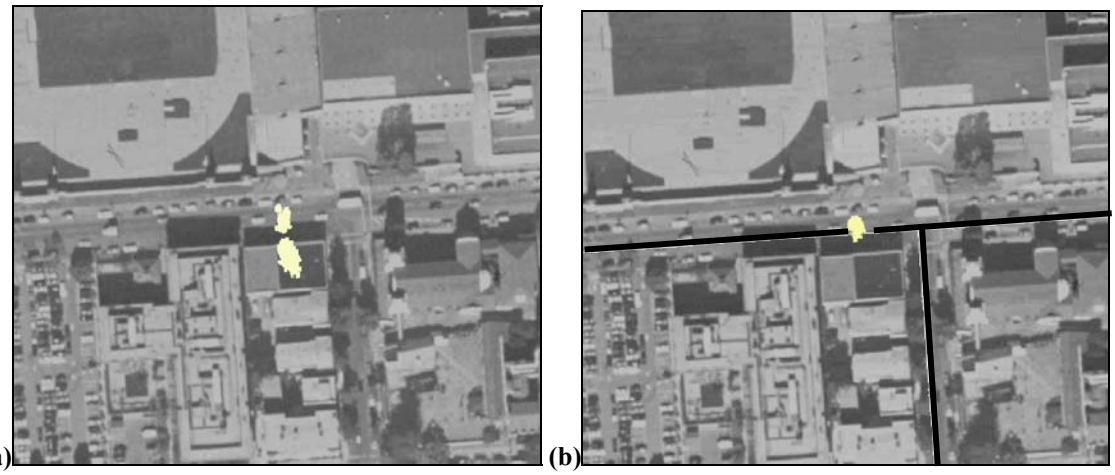
8.3.3 Monte Carlo Localization

In order to track the vehicle in the aerial image more reliably, we used the robust Monte-Carlo-Localization described in Section 5.5. We initialized a set S_0 of N particles by distributing them uniformly within an interval $[\pm\Delta x,\pm\Delta y,\pm\Delta\theta] = [\pm 10m,\pm\Delta 10m,\pm\Delta 10^0]$ around the selected starting position in the aerial image. Then, we apply the three phases motion, perception, and resampling iteratively for each step k, in order to arrive at a series of sets S_k, as shown in Figure 8-7. Superimposed on the edge map, Figure 8-7(a), Figure 8-7(b), and Figure 8-7(c) show the particle set S_k at iterations 0, 30, and 100 respectively. As seen, the blob of particles moves correctly along the path traversed during data acquisition.

Figure 8-7: Sets of particles (black) overlaid over aerial edge map (gray) (a) Initial uniform distribution S_0; (b) set S_{30} after 30 iterations of motion and perception; (c) set S_{100} after 100 iterations of motion and perception.

In residential areas with many trees, both false edges in the aerial image and increased inaccuracy in the scan-to-scan matching process contribute to a large position uncertainty. Therefore, the number N of used particles has to be quite large in order to represent the extended belief densely enough. In our experiments, we found out that to recover the entire path reliably, a number of N>120,000 particles is needed; this is about two orders of magnitude more than the "optimal" particles set size of 1000 to 5000, which Fox reported in [Fox et al., 2000] for a much smaller indoor environment and a hand-made edge map. However, if we opt to use the additional information a digital roadmap can provide, and constrain possible positions of particles to locations nearby roads, we can significantly reduce the necessary amount of particles.

Using the modified edge map shown in Figure 5-13, where a 25 meters wide strip around roads has been marked, we prohibit the selection of all "off-road" particles by assigning a zero importance during the resampling process. Hence, particles are not "wasted" on obviously incorrect locations and therefore much fewer particles are needed, while the probability distribution of the belief near the roads is still represented appropriately. In Figure 8-8, we compare the belief computed with N=200,000 particles without restrictions versus the belief computed with N=10,000 particles only, but restricted within the 25 meter wide strip around the roadmap we have marked in the edge map. As seen, the spread of the particles is significantly reduced in Figure 8-8(b). We have found that with the road-area restricted edge map, the amount of necessary particles to recover the path reliably drops to less than 10,000, thus decreasing computational time greatly.



(a) **(b)**

Figure 8-8: Restricting particles to locations near roads; belief computed with (a) N=200,000 particles without restrictions, (b) N=10,000 particles restricted within a 25 meter wide strip around the roadmap (black)

With the methods developed in section 5.5.2, we compute from each set S_k an intermediate global pose estimate $(\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$ by considering only particles which have descendants 50 resampling generations later, and we distribute global corrections among the relative steps. Concatenating these corrected steps, we obtain a final path registered with the aerial photo. Figure 8-9 shows the laser scan points drawn for each intermediate

position in black, superimposed with the edge image. It can be seen that these scan points match with edges of the aerial image in most cases, as shown in the two detail views in the lower left and upper right. The area with the most position uncertainty and thus the least match is a suburban area shown in the upper left detail view, where the algorithm faces many false edges corresponding to shadows of trees, and hence the ground based scans and aerial edge map are simply different. Despite this temporary loss of accuracy in position estimation, the algorithm recovers as soon as distinct features become available again. We have found that MCL is a very robust approach for recovering the driven path.

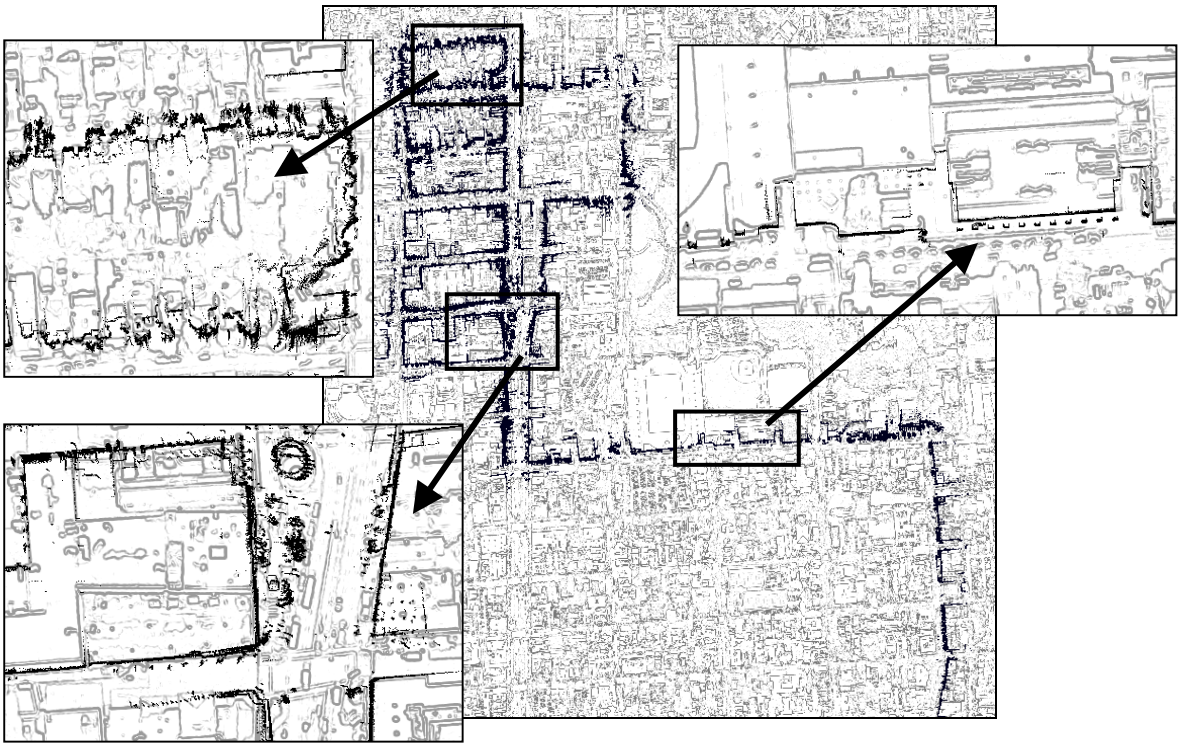


Figure 8-9: Scan points drawn for MCL-corrected path

Using an aerial photo, however, has two potential disadvantages: for the first, strong edges in the aerial photo might not be at their correct 2D location, since they may result from perspective-shifted building tops, or simply from intensity discontinuities only. Hence, even if the scan points match the global edge map, the position could be erroneous in regard to the true geographical location. For the second, no altitude information can be recovered, and hence the approach is restricted to flat urban areas. In the next section, we explore the use of an edge map from airborne laser scans instead of an aerial image in order to overcome both problems.

8.4 MCL Based on Airborne Laser Scans

Airborne laser scans of Berkeley were acquired in conjunction with Airborne 1 Inc., Los Angeles, CA, in about one hour acquisition time. The entire data set consists of 48 million scan points, which have an accuracy of 30 centimeters in horizontal and vertical direction and a raw spot spacing of 0.5 meter or less, and both the first and the last pulse of the returning laser light is measured. Choosing a grid spacing of 0.5 by 0.5 meter and applying the resampling technique described in Section 7.1, we obtain a Digital Surface Model as shown in Figure 8-10. Again, we need an edge map in order to apply the global correction procedure from Chapter 5. Therefore, we compute this edge map from the DSM by using the discontinuity filter proposed in Section 5.2.2, as shown for downtown Berkeley in Figure 8-11. We also compute a corresponding DTM containing the terrain altitude with the method we have described in Section 5.2.2.

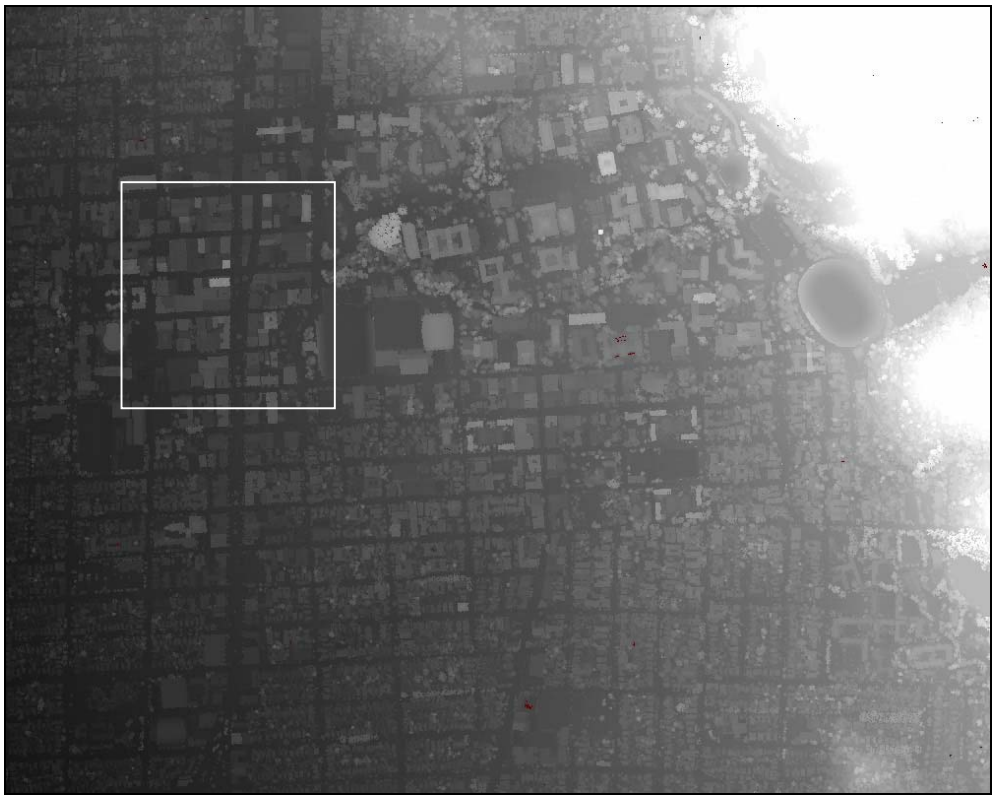


Figure 8-10: Digital Surface Model of Berkeley, encoded as gray image; the white rectangle marks the downtown area shown in the next figure

Figure 8-11: Edge map from DSM for downtown Berkeley area

Both the DSM and the corresponding edge map are in fact orthogonal projections onto the ground and are hence metric even for hillside areas, in contrast to the edge map from the aerial photo in the previous section. Therefore, we can use it to correct the global pose of the entire 10.2-kilometer path, including the first part in the hills. We have applied the same procedure as for the edge map from aerial images, i.e. selecting the starting position in the edge map, applying iteratively Monte Carlo Localization for each relative step k to obtain a particle set S_k, and computing a corrected global pose from each S_k.

We have found that despite its lower resolution, the edge map from the DSM is far superior to the edge map from the aerial images, due to the absence of false edges and perspective shifts. For the 1-foot aerial images, the uncertainty was enormous at some locations, and without digital roadmap, it was necessary to utilize 120,000 particles to approximate the spread-out probability distribution appropriately and track the vehicle reliably. In contrast, for the edge map derived from airborne laser scans, the spread of the particle set was extremely low throughout the entire computation, and we have found that the vehicle could easily be tracked with as little as 1000 particles. It is recommendable though to use more particles in order to minimize noise in the global pose estimates. Figure 8-12 shows an example particle sets S_k, computed during Monte-Carlo-Localization with 5000 particles. It can be seen that the spread of the particles and hence the uncertainty of the global pose estimation is small even in residential and hillside areas.

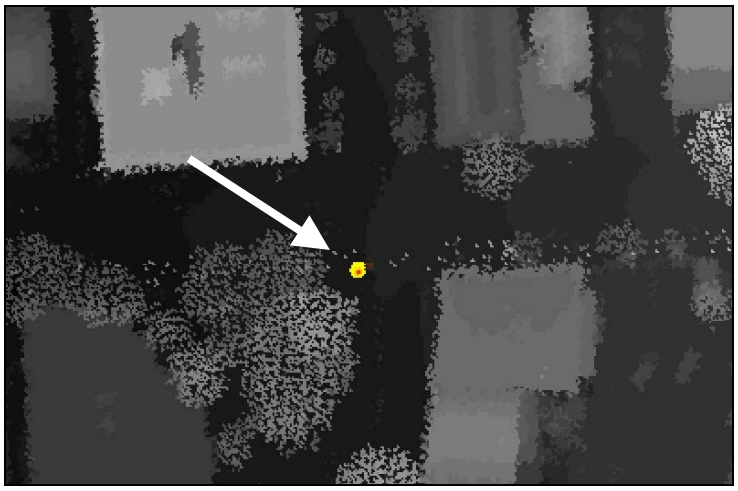


Figure 8-12: Set of particles (yellow/red) overlaid over DSM (gray)

After computing intermediate global pose estimates from the particle sets, we have applied the global correction according to Section 5.5.2 in the following manner: First, we calculated the yaw angle difference between initial path, as shown in Figure 8-13(a), and we correct the yaw angles according to the smoothed difference. Then, we have recomputed the path, and calculated the x and y differences to the intermediate global poses, as shown in Figure 8-13(b). Similarly, we corrected the x and y coordinates according to the smoothed x and y differences, and obtained the final path. In both figures, the curves around the horizontal axes are the differences after correction, and as seen, only the high-frequency components are kept. We found that errors in the yaw angles are by far more significant for the global position than errors in the x,y estimates: while the initial path is off by several hundred meters, the difference between the yaw-angle corrected path and the intermediate global positions is only a few meters, as seen Figure 8-13(b).

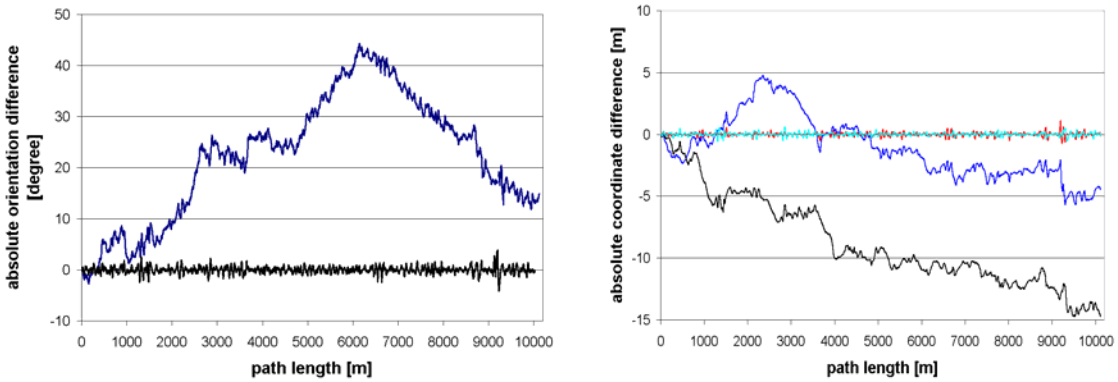
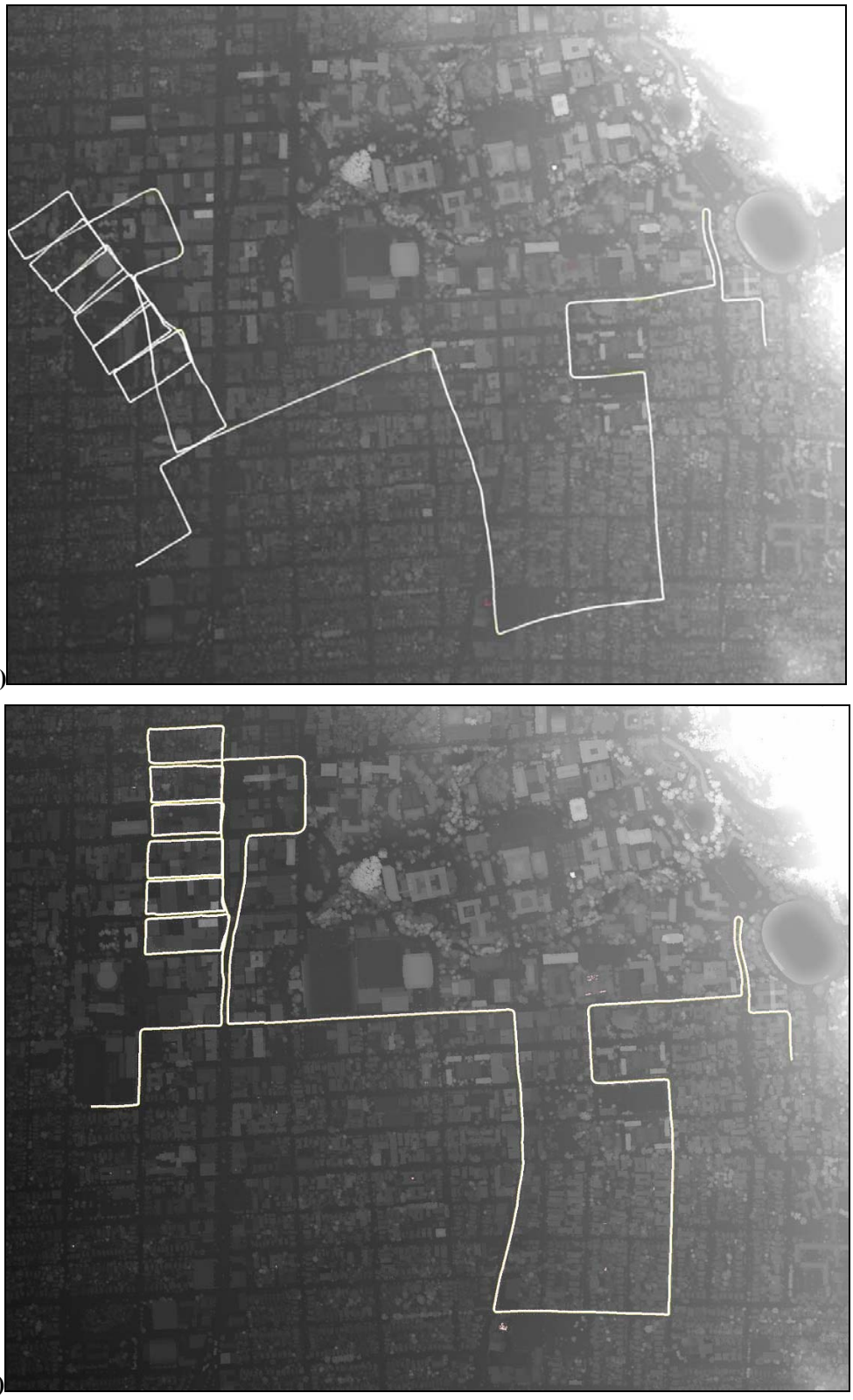


Figure 8-13: Global correction along the traveled path; (a) yaw angle difference between initial path and global estimates before and after correction; (b) differences of x and y coordinates before and after correction. In both diagrams, the differences after corrections are small (curves close to the horizontal axis).



(a)

(b)

Figure 8-14: Entire traveled path superimposed on top of the DSM; (a) initial path from scan-to-scan matching; (b) path corrected with MCL.

In Figure 8-14, the entire 10.2 km path including the hillside segment is drawn in white and superimposed on top of the DSM. Figure 8-14(a) shows the initial path obtained from scan-to-scan matching, and Figure 8-14(b) the path corrected with MCL, using the edge map from the DSM. As seen, the initial path does not match the DSM, whereas the MCL corrected path fits well to it. Accordingly, for the corrected path, the ground-based horizontal scan points match the depth discontinuities in the DSM, as shown in the close-up view on the downtown area in Figure 8-15. Thus, the ground-based data is registered with the airborne scans.

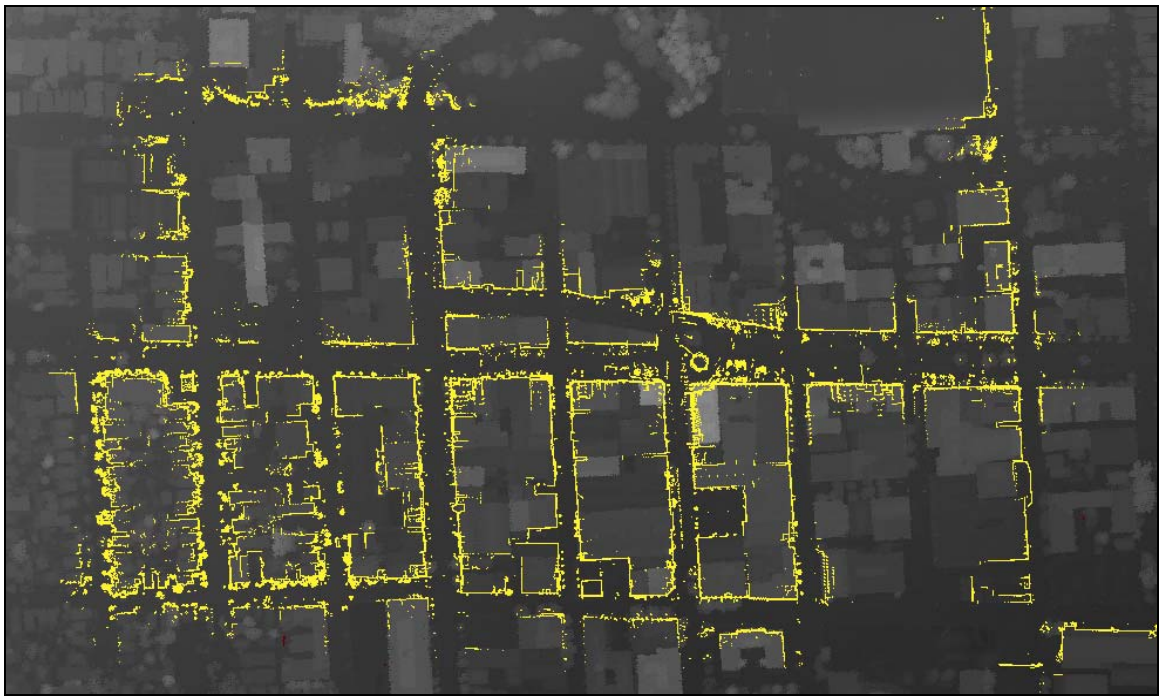


Figure 8-15: Horizontal scan points for corrected path superimposed on top of the DSM.

Using the DTM computed from the DSM, we assign each 3-DOF pose the altitude as the z-coordinate and the slope as the z-derivative in driving direction, or equivalently, the more commonly used pitch angle as the inverse tangent of the slope. Figure 8-13(a) shows the z-coordinate and Figure 8-13(b) the pitch angle and hence the incline during the drive; clearly visible is the incline from our higher starting position near the Berkeley Hills down towards the San Francisco Bay, as well as the ups and downs on this incline while looping around the downtown blocks. While the pitch angle reaches 6 degrees and more in some parts of the hillside drive, it is at most about 2 degrees for the more level downtown blocks.

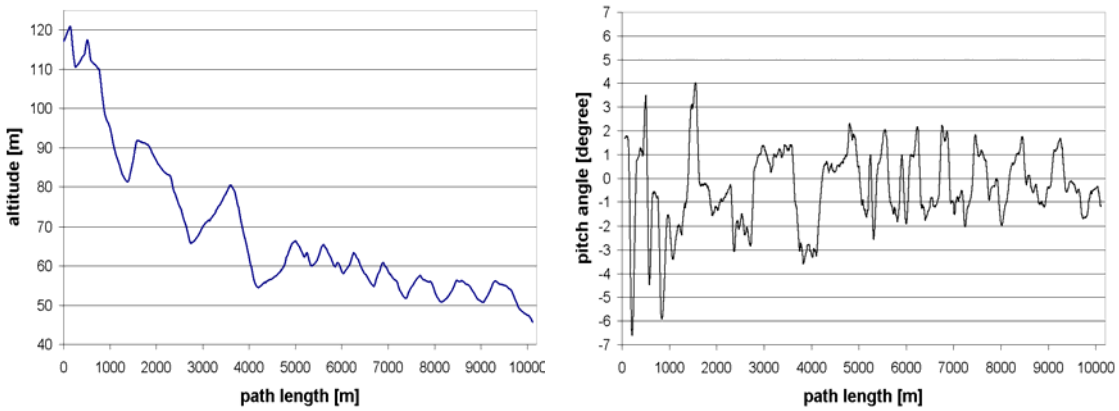


Figure 8-16: Assigned z coordinates and pitch angle

8.5 Facade Model Generation

After determining the vehicle's pose accurately for each scan as described in the previous section, it is straightforward to transform the vertical 2D scans into 3D scan points. We now apply the framework of data processing techniques introduced in Chapter 6 in order to create visually appealing facades completely automatically. In the following, we apply our geometry reconstruction techniques to the entire 6769 meters long path segment starting from Telegraph Avenue to the end. For our geometry processing, we utilize the 6-DOF path corrected with MCL using the edge map from the DSM, since it contains also altitude and incline information. We can subsequently apply automated texture mapping to the downtown blocks, for which we captured all facades accessible from the roads and for which we have recorded color images.

Applying the path splitting techniques described in 6.1 results in 73 quasi-linear path segments, as shown in Figure 8-17 overlaid with the digital roadmap. There is no need for further manual cutting, even at Shattuck Avenue, where the "Manhattan geometry" of Berkeley is not preserved.

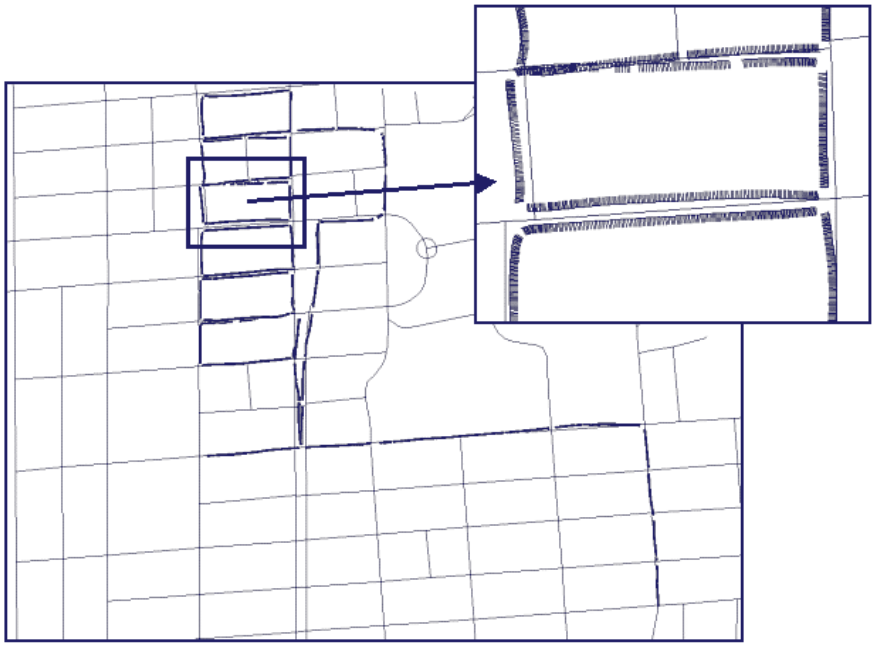


Figure 8-17: Entire path after split in quasi-linear segments.

We have applied the postprocessing methods to all 73 segments, and performed an evaluation of the obtained results. As we do not have the ground truth to compare with, and as our main concern is the visual quality of the generated model, we have generated two facade meshes for each of the 73 segments for comparison: the first mesh is obtained by directly triangulating the raw scans, and the second one from the depth image to which we have applied the postprocessing steps described in Chapter 1. Manually inspecting the results, we have subjectively classified the degree to which the proposed postprocessing procedures have improved the visual appearance. The classification categories are "significantly better", "better", "same", "worse" and "significantly worse". In Figure 8-18, we show a comparison of some examples for meshes generated directly from the raw scan points (right) and meshes generated after the postprocessing steps (left), together with the corresponding classifications we assigned. As seen, except for pair "f", the proposed postprocessing steps result in visually more pleasing models.

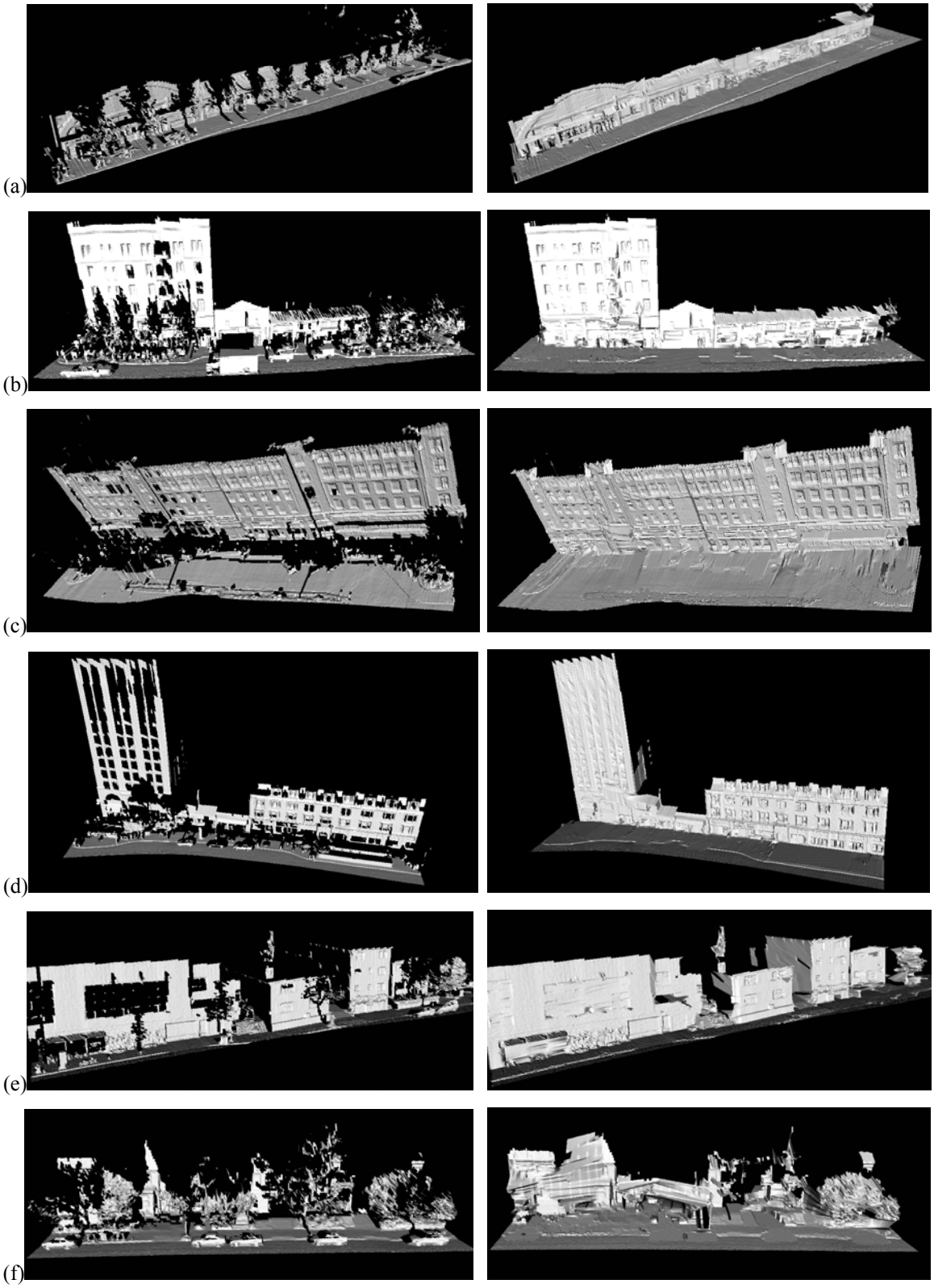


Figure 8-18: Generated facade meshes, left side original, right side after the proposed foreground removal and hole filling procedure. The classification for the visual impression is "significantly better" for the first four image pairs, "better" for pair e and "worse" for pair f.

The evaluation results for all 73 segments with and without the postprocessing techniques described in Chapter 6 are shown in Table 1. Even though 8 % of all processed segments appear visually inferior to the original, the overall quality of the facade models is significantly improved.

Significantly better	35	48 %
Better	17	23 %
Same	15	21 %
Worse	5	7 %
Significantly worse	1	1 %
Total	*73*	*100%*

Table 1: Subjective comparison of the processed mesh vs. the original mesh for all 73 segments.

We have found our processing methods to work well in the downtown areas, where there are clear building structures and few trees. The important downtown segments are in most cases ready to use and do not require any further manual intervention. However, in residential areas, where the buildings are often almost completely hidden behind trees, it is difficult to accurately estimate the geometry. Hence, the few problematic segments all occur in residential areas, consisting mainly of trees. The tree detection algorithm in section 6.4 classifies 10 segments as "critical" in that too many trees are present. Pair f in Figure 8-18 is one of these segments, and hence should be omitted or left "as is" rather than processed. All 6 problematic segments corresponding to "worse" and "significantly worse" rows in Table 1 are among them, yet none of the improved segments in rows 1 and 2 are detected as critical. This is significant because it shows that (a) all problematic segments correspond indeed to regions with a large number of trees, and (b) they can be successfully detected and hence not be subjected to the proposed steps. Table 2 shows the evaluation results if only non-critical segments are processed. As seen, the postprocessing steps described in Chapter 6 together with the tree detection algorithm improve over 80% of the segments, and never result in degradations for any of the segments.

Significantly better	35	56 %
Better	17	27 %
Same	11	17 %
Worse	0	0 %
Significantly worse	0	0 %
Total	*63*	*100%*

Table 2: Subjective comparison of the processed mesh vs. the original mesh for the segments automatically classified as non-tree-areas.

Note that the evaluation of our technique is based on comparing the non-textured geometry. In a comparison in which both models are texture mapped, the processed mesh is even more likely to be visually superior to the original, since texture distracts the

human eye from geometry imperfections such as those potentially introduced by hole filling algorithms. With the automated foreground marking procedure described in Section 6.6, the facade meshes can be texture mapped. In Figure 8-19, we compare the two facade meshes for downtown city block side shown in Figure 8-18(d) after texture mapping. The upper mesh is generated without and the lower mesh with the postprocessing procedure, respectively. As seen, the visual difference between the two meshes is striking, for the reasons described above. Note the facade area occluded by the two trees on the left side of the original mesh has been completely filled with geometry and texture mapped from oblique camera views as much as possible. Nevertheless, a few triangles are not visible in any camera image and therefore left untextured, possibly to be subjected to a texture synthesis algorithm in future work.

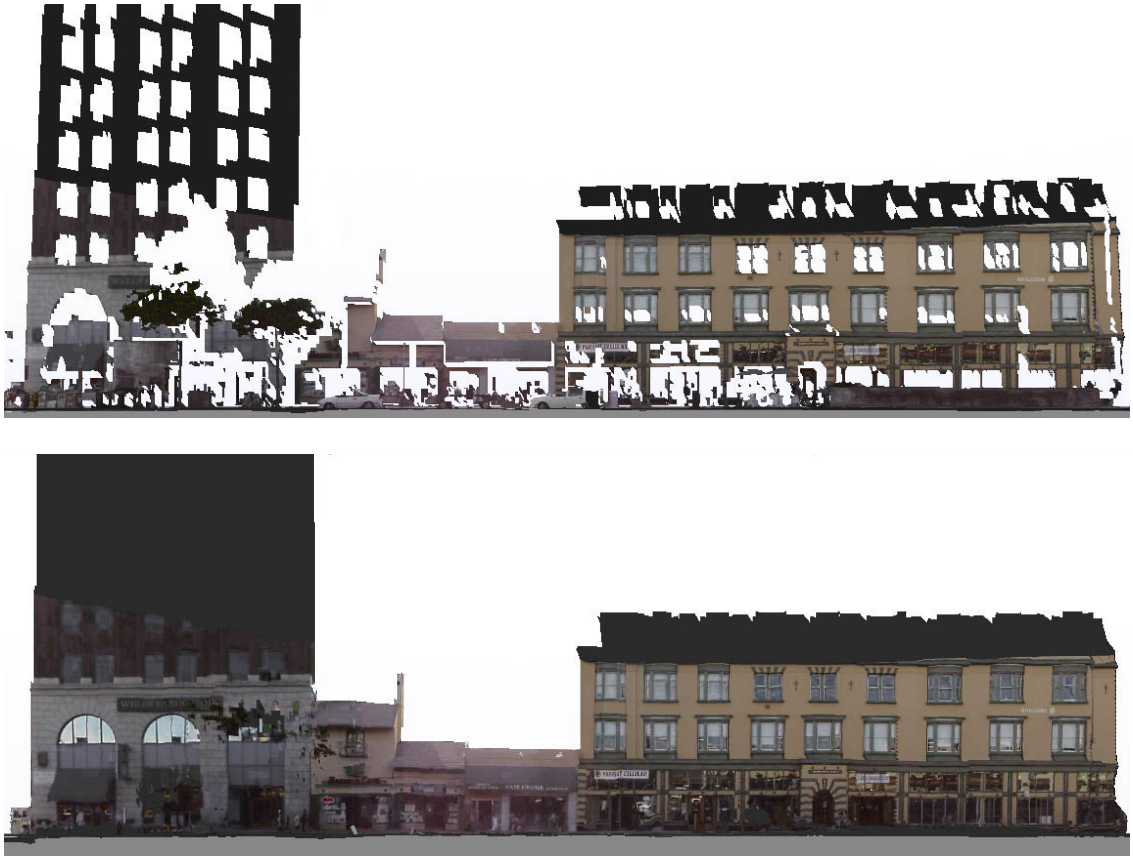


Figure 8-19: Textured facade mesh without (top) and with (bottom) processing.

For 12 out of the 73 segments, additional 3D vertices derived from stereo vision techniques are available. Sorting in these 3D points into the layers according to section 6.4 does fill some of the holes. For these specific holes, we have filled the holes prior to our processing based on stereo vision vertices and compared the results with those solely based interpolation as in Table 1 and Table 2. We have found no substantial differences,

suggesting that our processing and interpolation scheme alone yields already reasonable results. Often, the interpolated mesh vertices even appear to be slightly more accurate than the stereo vision based vertices as they are less noisy, and hence we found no reason of additional exploiting stereo vision data for our model generation. Figure 8-20(a) shows an example before processing, and Figure 8-20(b) shows the tree holes completely filled in by stereo vision vertices. As seen, the outline of the original holes can still be recognized in Figure 8-20(b), whereas the points generated by interpolation alone are almost indistinguishable from the surrounding geometry, as seen in Figure 8-20(c).

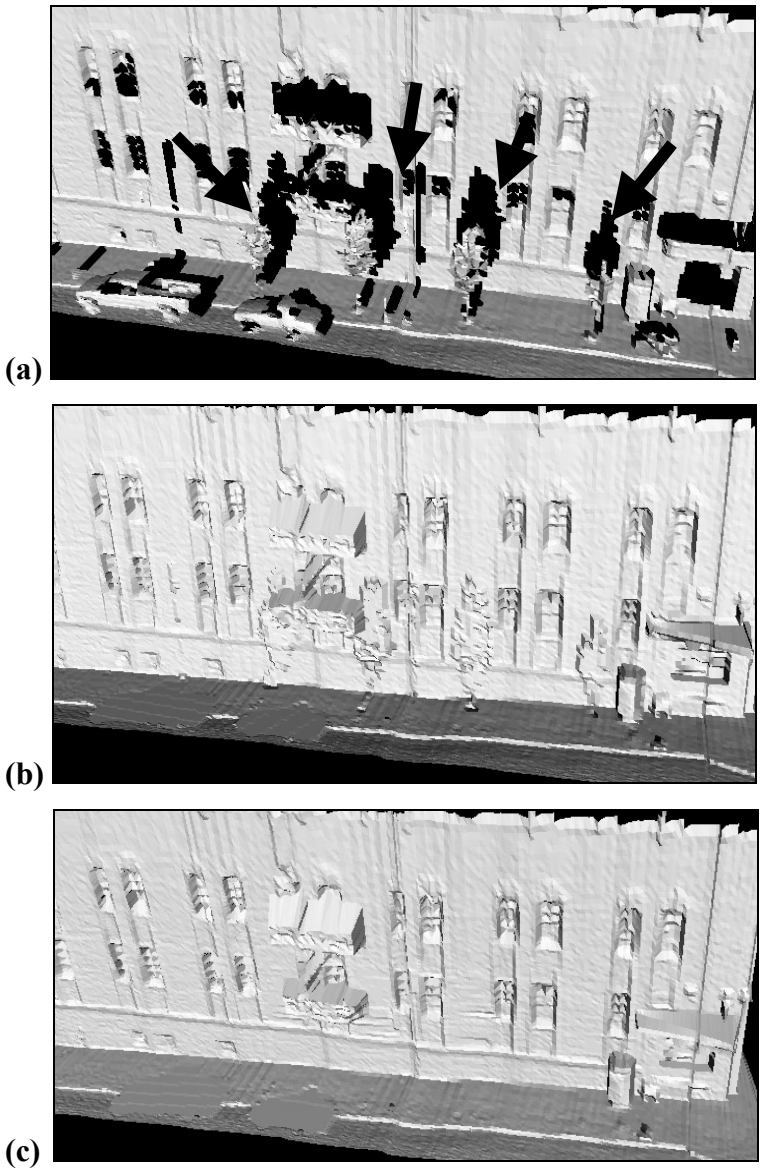


Figure 8-20: Hole filling (a) original mesh with holes behind occluding trees; (b) filled by sorting in additional 3D points using stereo vision; (c) filled by using the interpolation techniques of section 6.5

Combining all individually processed segments in the common global coordinate system, we obtain facade meshes for the entire traveled path starting from Telegraph Avenue, as shown in Figure 8-21. For better visualization, these facades are superimposed over an aerial image.

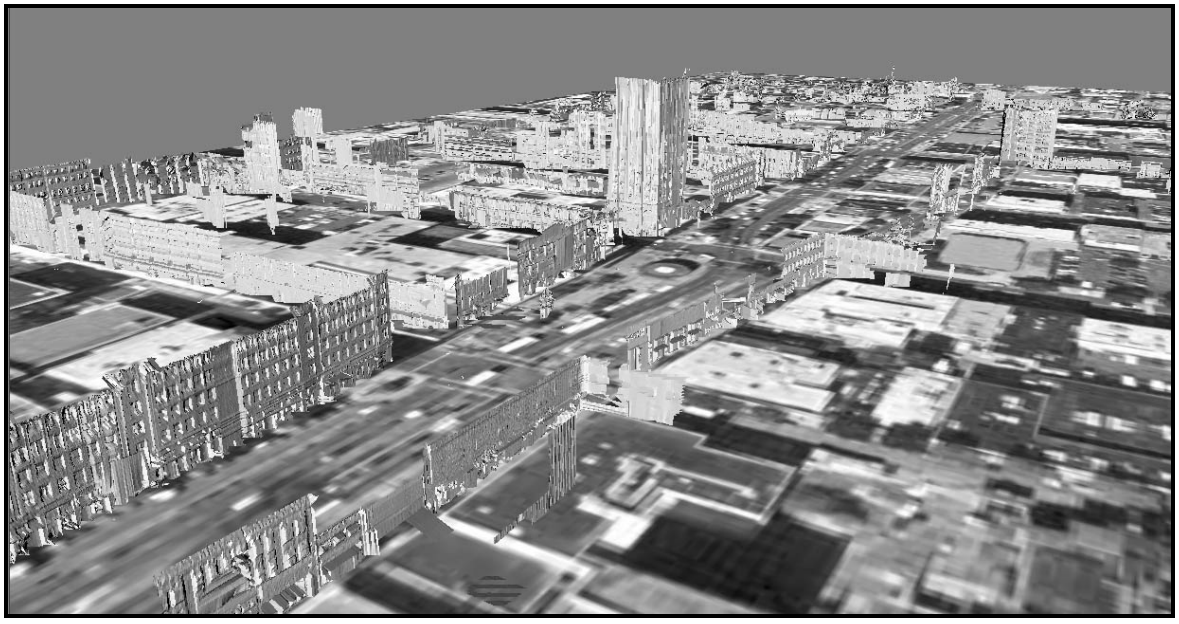


Figure 8-21: Non-textured facade models for the entire path, overlaid on top of an aerial photo.

For the looped downtown blocks, we have camera images available and can hence apply the automated texture mapping procedure described in section 6.6, which is capable of handling occlusions and determining the image areas with a direct view on the facades behind. We texture map the facade meshes completely except the upper parts of tall buildings, which were out of the limited field of view of the camera during data acquisition. A bird's eye view over the texture mapped downtown facade models is shown in Figure 8-22, and close-up views are shown in Figure 8-23 and Figure 8-24.

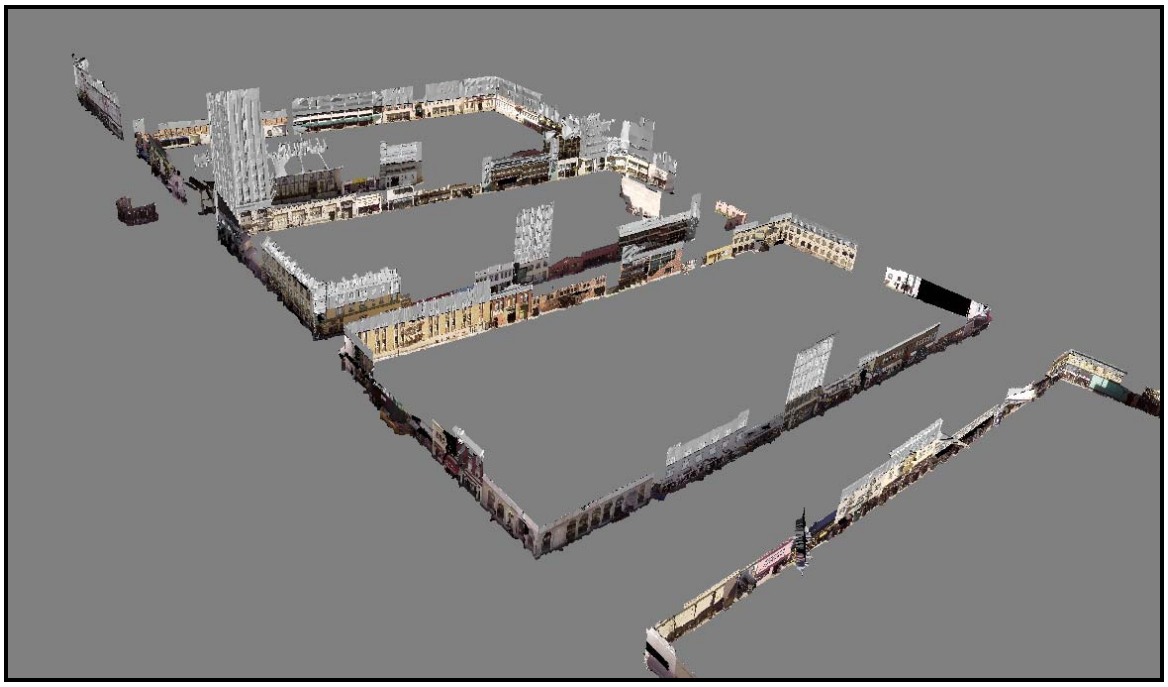


Figure 8-22: Bird's eye view on the texture mapped facade models for the downtown blocks

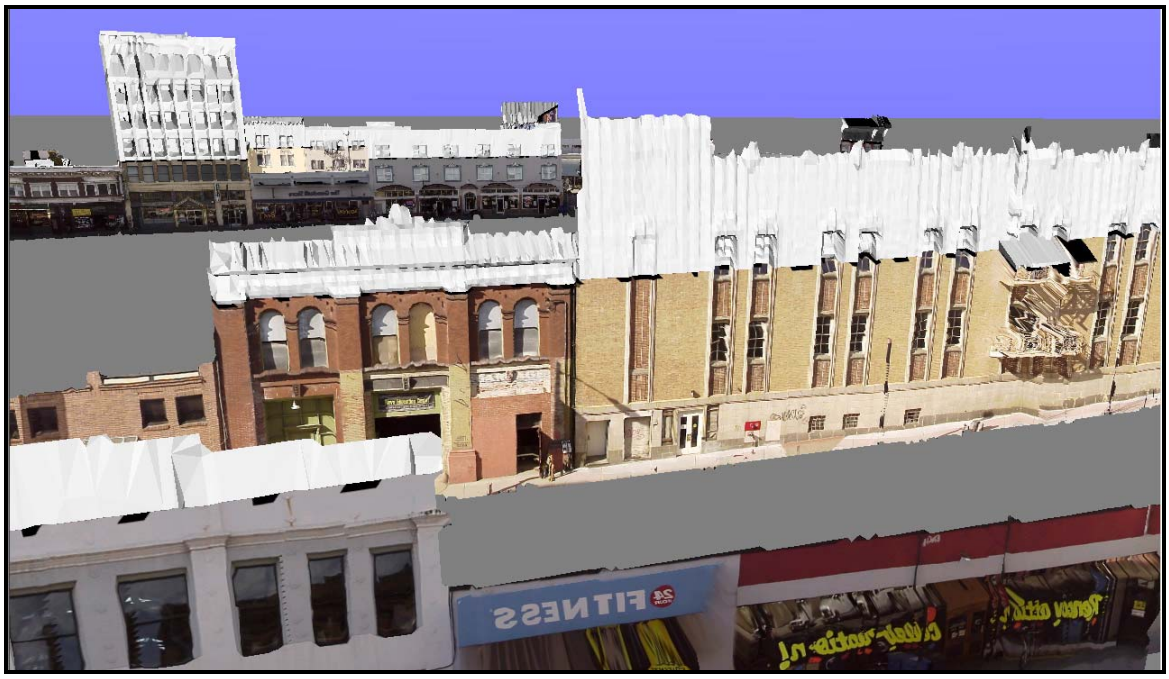


Figure 8-23: Close-up view on the ground-based facade models, seen from the backside of an Addison Street facade.

Figure 8-24: Close-up view on the ground-based facade models, seen from Center Street.

According to section 6.7, we optimize the facade meshes for rendering by creating multiple levels of details; specifically, we generate for each an atlas as an efficient texture representation and reduce the number of triangles using the Qslim mesh simplification tool. Then, we further subdivide the meshes along vertical cutting planes and combine all sub-meshes in a hierarchical, 3-LOD scene graph. This enables us to render the texture mapped facades interactively for the entire downtown path with a standard VRML viewer, as seen in the screenshot in Figure 8-25.

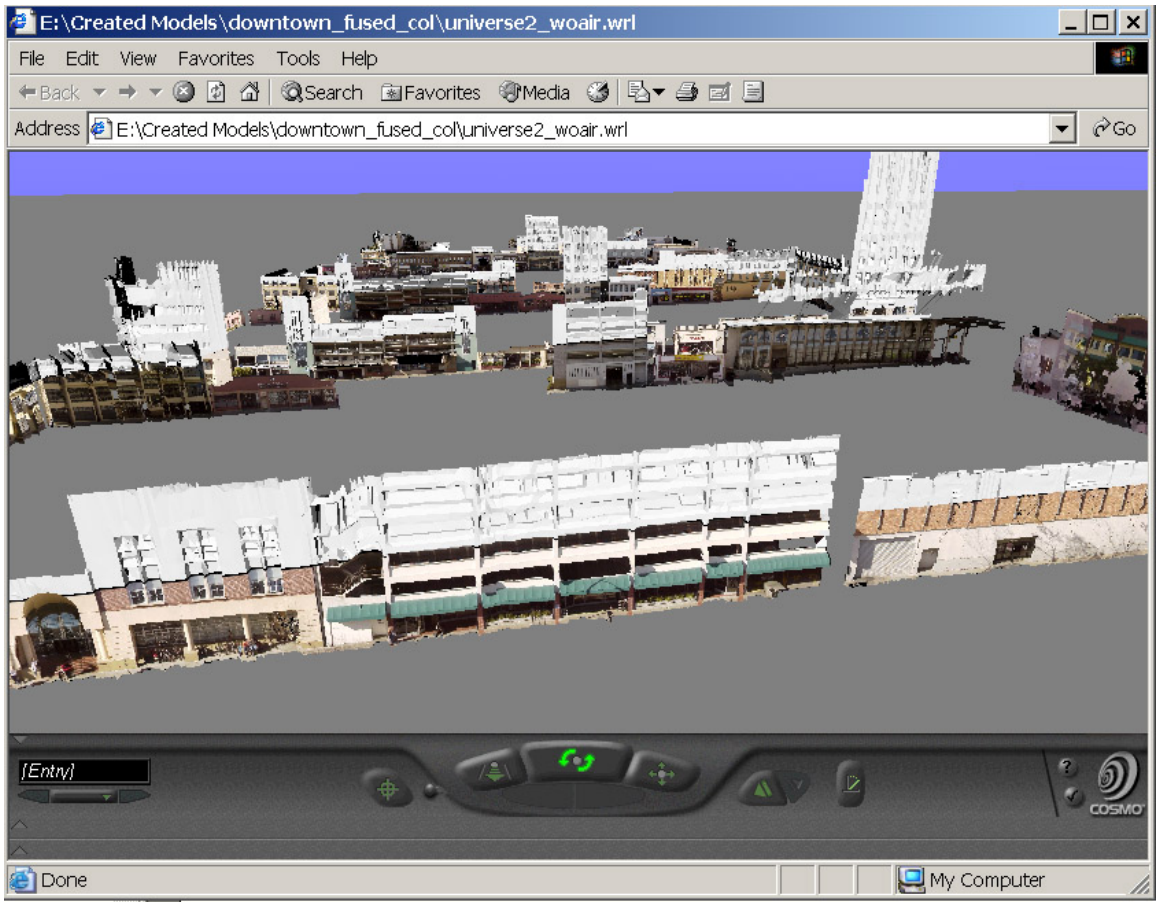


Figure 8-25: Viewing the texture mapped model with a standard web-based VRML browser, in this case Computer Associates' Cosmo Player.

8.6 Airborne Modeling

According to section 7.2, we have computed an airborne surface mesh for the downtown Berkeley area by (a) resampling aerial laser scan points to a DSM, (b) processing the DSM by flatten planar surfaces and straightening edges, (c) connecting vertices in the DSM, and (d) reducing the mesh to about 100,000 triangles per square kilometer by using qslim mesh simplification. Figure 8-26(b) shows the resulting surface mesh. For comparison purpose, Figure 8-26(a) shows the surface mesh as obtained without the DSM processing steps; as seen, the processing improves the visual appearance of the mesh significantly. In addition, during a helicopter flight, we have acquired 12 aerial color images of the Berkeley area at oblique angles. In about an hour of manual work, we have registered these images by selecting correspondence points between the images and the DSM, and automatically solving for the camera pose. Knowing the camera pose, the corresponding image location for each surface mesh vertex has been computed, the optimal image for each triangle has been selected by taking into account resolution, normal vector orientation, and occlusions, and the mesh has been texture mapped

accordingly. Figure 8-27(c) shows the airborne model from Figure 8-26(b) after texture mapping with 12 aerial images.

It is difficult to evaluate the accuracy of this airborne model, as no ground truth with sufficient accuracy is readily available, even at the city's planning department. It can be noted though that by removing small features on building tops, we have admittedly sacrificed geometric accuracy for the sake of visual appearance. However, while some details are actually missing in the geometry, they visually appear to be present due to the texture-mapped aerial imagery. Thus, by creating a false impression of geometric detail, our approach combines elements of model-based and image-based rendering. While this is undesirable in some applications, we believe it is appropriate for the purpose of interactive visualization.

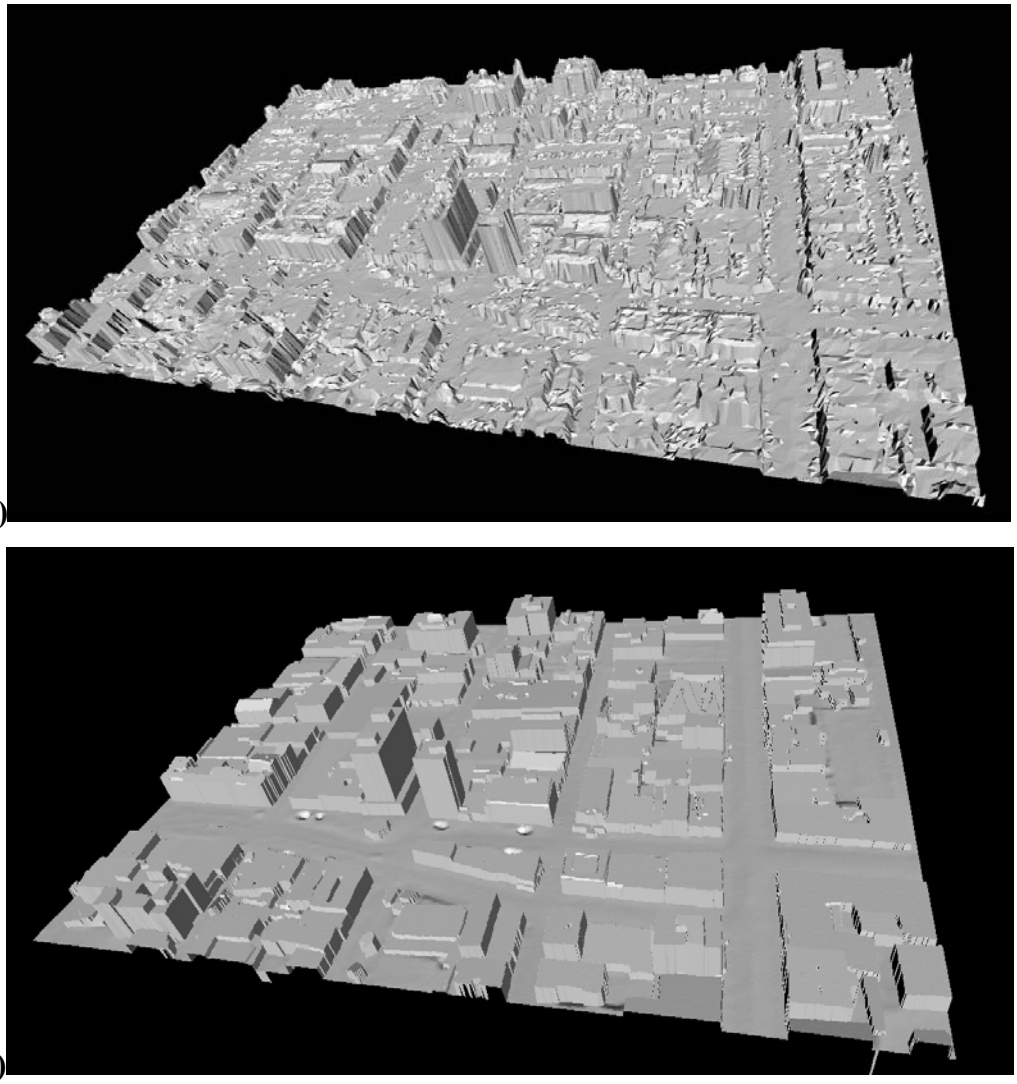


Figure 8-26: Airborne model for downtown Berkeley; (a) original DSM directly triangulated, (b) triangulated after DSM postprocessing.

Figure 8-27: Airborne model for downtown Berkeley after texture-mapping with 12 aerial images.

8.7 Model Merging

Monte-Carlo-Localization and pose correction have globally adjusted the path to fit the edges of the DSM. Thus, as a great advantage of our specific localization approach, the ground-based models and the airborne surface mesh derived from the DSM are automatically registered with each other. However, if no model merging is performed and the models are simply overlaid in the same coordinate system, it is random which mesh is on top and thus visible, and as seen in Figure 8-28 and indicated by arrows, the low-resolution, coarse airborne mesh covers at numerous locations the high-quality facade models.

We perform the model merging methods described in Section 7.3, i.e. we mark DSM cells corresponding to ground-based facade vertices and foreground objects, regenerate the surface mesh, and create a blend mesh to fill gaps and smooth the transition between the two meshes. Figure 8-29 illustrates the visual superiority of the highly detailed ground-based facade models for walk-thrus: while the facades on the right street side originate from the airborne surface mesh, they are on the left side replaced by the highly detailed ground-based facade model. As seen, the inserted ground-based facades appear significantly more suitable for a walk-thru. Figure 8-30 shows the resulting combined model for the looped downtown Berkeley blocks viewed while walking or driving, Figure 8-31 shows the same model as seen from a building top, and Figure 8-32 shows the model as it occurs in a Bird's eye view.

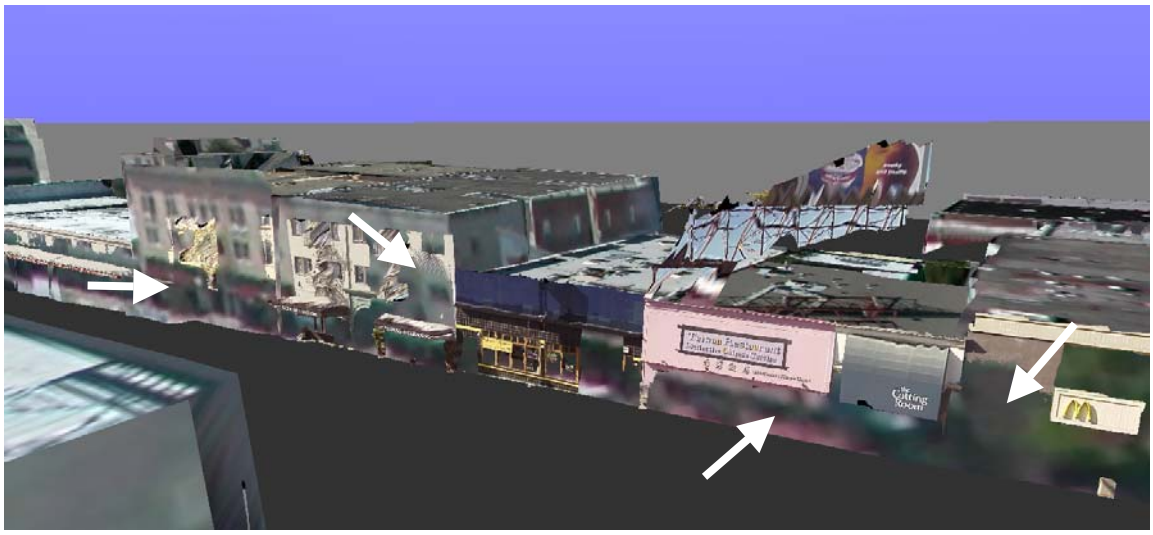


Figure 8-28: Ground-based models and airborne surface mesh overlaid on top of each other, without applying model merging steps. While the two meshes are registered with each other, the coarse airborne triangles cover the high-resolution facade models in numerous locations, e.g. where indicated by the white arrows.

Figure 8-29: Comparison of walk-thru view on facades from ground based versus airborne acquisition; while the facade on the right street side originates from the airborne surface mesh, it is on the left side replaced by the highly detailed ground-based facade model.

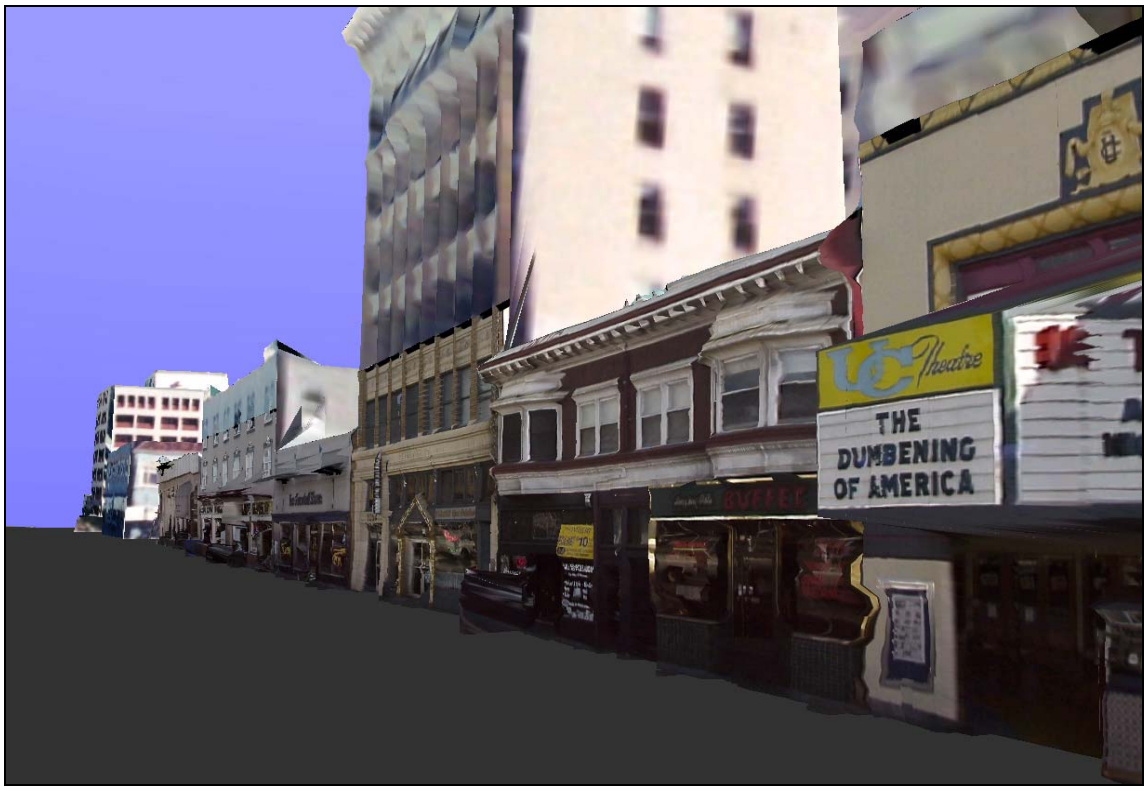


Figure 8-30: Walk-thru view of the merged model

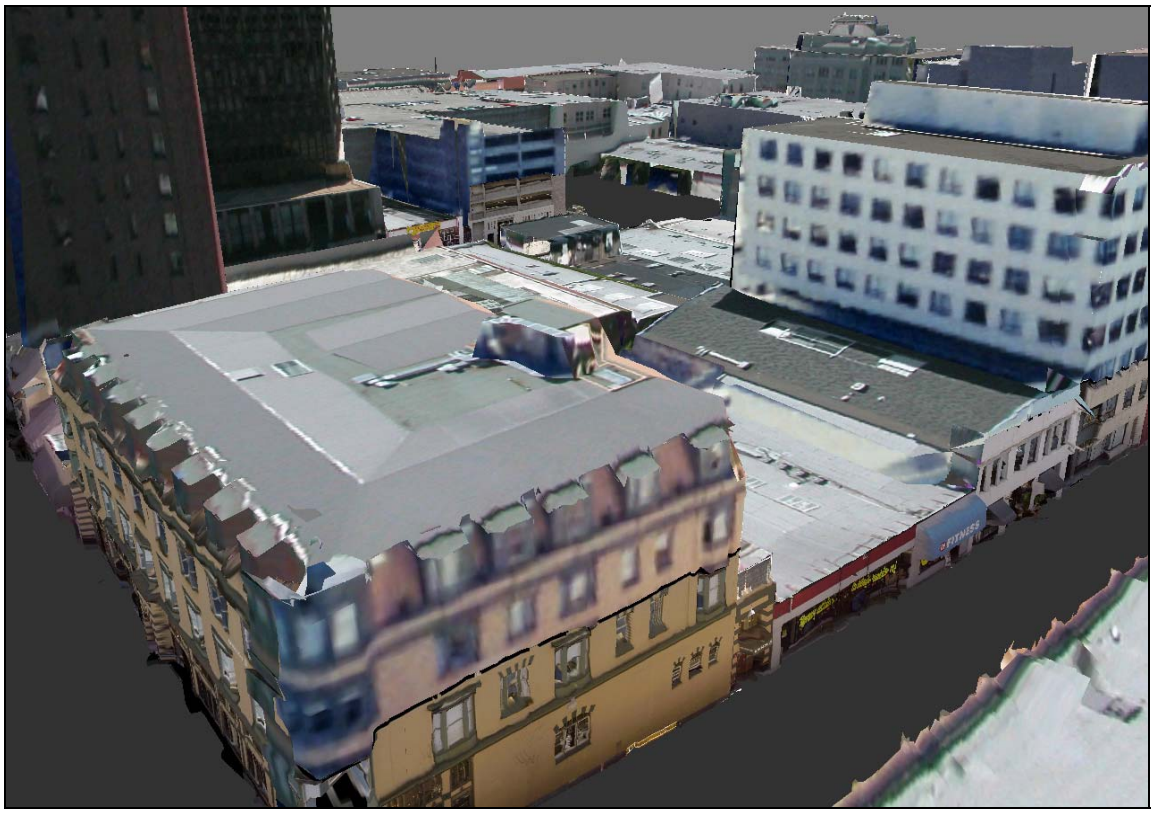


Figure 8-31: Virtual view from a building top of the merged model.

Figure 8-32: Bird's eye view of the merged model.

8.8 Performance and Complexity

Besides automatism and photo-realism, the scalability to large environments is one of our key objectives. We have applied our methods to a substantial urban area, and this section is dedicated to analyzing computational performance and scalability for the processed data and beyond it. Ultimately, our methods should be applicable to extremely large urban environments with up to tens of square miles, e.g. the metropolitan Los Angeles or the San Francisco Bay Area.

The proposed acquisition and processing scheme utilizes a large number of various individual algorithms, taking different input and output data and thus having different individual complexity measures. However, scalability to large environments means the computational complexity as a function of the covered area size. Since we can fairly assume that area size, number of contained facades, and path length are approximately proportional, the crucial question is how the complexity increases with the length of the driven path, and we analyze this complexity in the following:

First, the data acquisition time is linear in path length, and so is the amount of acquired data. While practically the maximum path length is limited by the size of the storage media, it is trivial to overcome this potential bottleneck simply by adding additional hard disk drives. Second, scan-to-scan matching, initial path computation, and MCL correction

are linear in path length, and similarly edge map generation from both aerial photo and DSM is linear in area size. Here, the usage of a global edge map for the localization turns out to be enormously advantageous, since we do not have to perform the $O(n^2)$ data cross consistency matching necessary in some previous approaches. Third, the ground based data processing is linear in path length: while some of the functions such as marking foreground in the images for texture mapping show $O(n^2)$ complexity with the size of the path segment in our implementation, this does not have an effect on the overall scalability, since the segment size is confined to a maximum length by the path segmentation techniques. Fourth, due to the O(n) behavior of the qslim mesh simplification, the airborne surface mesh generation is also linear in area size, and similarly the DSM marking and blending is linear in path length. Hence, the proposed automated model generation procedure is completely linear in area size and path length, and thus easily scalable to large environments. In fact, the average time per area for the diminutive manual interaction practically even decreases with larger areas, since the two necessary manual steps besides driving, i.e. entering the starting position for the MCL localization and selecting correspondence points for the airborne texture mapping, are one-time tasks independent on the covered area size.

Table 3 shows the processing time for the vehicle localization, i.e. the path computation and global correction based on the edge map from the DSM, for the entire 10,2 km/37 minutes drive on a 2 GHz Pentium 4 PC. The computationally most expensive part is the accurate scan matching process, with about two thirds of the total computation time of roughly 4 hours. Since the computation time is proportional to the path length, one can figuratively consider our "localization speed" to be about 43 meters path per minute.

Processing times to localize the vehicle for the entire 10,2-km/ 37-minutes drive	
Data conversion	32 min
Scan matching and initial path computation	148 min
Monte Carlo Localization (with DSM and 5,000 particles) and global correction	55 min
Total localization time	**235 min**

Table 3: Computation times for the vehicle localization, i.e. the complete path computation and global correction, for the entire 10.2 km path traversed in 37 minutes driving time.

For the 3043 meters/11 minutes downtown part of our drive, where we loop around the block between Shattuck Avenue and Milvia Street, we have performed the entire model generation procedure, including texture mapping and merging with the DSM, and have measured the computation times again on a 2 GHz Pentium 4 PC, as shown in Table 4. Although this data set consists of millions of scan points and a few thousand camera images, the processing time for the entire model generation is only about 2 ½ hours. Since again this computation time scales with path length and area size, respectively, one can figuratively consider our "total model generation speed" to be about 20 meter facades per minute.

Processing times for the 3043-meter/ 11-minutes downtown drive	
Data conversion	14 min
Scan matching and initial path computation	52 min
Monte Carlo Localization (with DSM and 5,000 particles) and global correction	18 min
Path segmentation	1 min
Geometry reconstruction	6 min
Texture mapping	27 min
Model optimization for rendering	19 min
DSM computation and projecting facade locations	6 min
Generating textured airborne mesh and blending	19 min
Total model generation time	**162 min**

Table 4: Processing times for the entire model generation of the downtown Berkeley blocks, acquired in a 3043-meter/ 11-minutes drive

Note that there is additionally the manual step of selecting correspondences for the registration of DSM and aerial image, necessary for texture mapping the airborne surface mesh. The selection of 8 to 10 correspondence points for each image took about an hour for a large area of Berkeley; since the area covered by these images is about 4 times larger than the downtown blocks, the manual selection time prorated to the downtown area is only about 15 minutes. All other steps of the airborne model generation and the entire ground-based model generation are completely automated and do not need manual intervention at any point.

As a summary for this section, it can be noted that our approach results in a highly detailed, photo-realistic 3D city model suitable for both virtual walk- and fly-thrus. In contrast to most other methods, the generation of this photo-realistic model is automated, and furthermore, our approach is extremely fast: the acquisition time for five downtown Berkeley blocks has been only 11 minutes, and the total processing time only 2 ½ hours, hence to the best of our knowledge outperforming all existing methods. Since the complexity of the developed approach is linear in area size, it is scalable and applicable to large environments.

9 Summary and Conclusion

This dissertation addresses the problem of acquiring photo-realistic models of urban environments for various virtual reality purposes, in particular for virtual walk-, drive- or fly-thrus. In photo-realistic walk- or drive-thru applications, an enormous level of detail at ground level is required. Existing semi-automated methods fall short of providing this level of detail in reasonable time and at affordable costs, since they are complicated and require substantial manual intervention. In this thesis, we propose an innovative approach to overcome these limitations by capturing highly detailed facade models from a ground-based view. The broad scope of the model acquisition problem requires an interdisciplinary solution, and accordingly, we employ methods from mobile robotics, optical measurement techniques, computer vision, and computer graphics. Our key design objectives are automatism, speed, scalability, and photo-realism: firstly, our approach is completely automated, i.e. no manual intervention is needed at any time. Secondly, both data acquisition and processing are extremely fast, potentially enabling the generation of a large-scale city model within hours. Thirdly, the complexity of the devised algorithms is linear, and computation time increases only proportionally to the covered area; thus, the algorithms are in principle extendable to arbitrary large areas. Fourthly, the created building models are visually as realistic as a photograph.

In our approach, we capture facade models from a ground-based view, while moving at normal speeds on public roads, with an acquisition vehicle equipped with laser scanners and a digital camera. Data is acquired continuously while driving by the buildings, rather than in a slow stop-and-go fashion, thus allowing the traversal of an entire city within a few hours. The collected data is processed offline, and 3D models of the building facades are generated completely automatically. Additionally, we show that it is possible to merge these facade models with data from an airborne view, in order to create a complete 3D model, containing facades as well as building tops and terrain shape.

More specifically, we have developed a sensor system and have mounted this system on a rack on top of a pickup truck. The 3D geometry of a city is acquired using a combination of two inexpensive 2D laser scanners mounted perpendicularly to one another, one horizontally and one vertically. The corresponding texture is acquired using a digital color camera. All devises are controlled by real-time software and synchronized by hardware signals; hence, it is possible to determine corresponding individual scans and images. In contrast to the 3D scan registration problem commonly occurring in approaches based on 3D scanners, we face a localization problem: in order to combine the individual scans and images to a model, the pose of the sensors during the acquisition has to be determined extremely precisely. We propose to solve this problem in an offline computation by first matching subsequent horizontal scans in order to obtain their relative pose and then by concatenating these relative poses to an initial path estimate. Because during longer periods of driving, small errors in the relative poses accumulate to form substantial errors in global pose, we devise, in addition, a global correction based on an edge map of the city area, which we derive either from aerial images or from airborne laser scans. Using probabilistic Monte-Carlo Localization, we are able to match

horizontal laser scans to the edge map and correct the path globally. Hence, we can assign a globally correct pose to each acquired laser scan and camera image.

We have developed an entire framework of completely automated algorithms in order to fuse the raw data to globally consistent facade models. First, we handle the large amounts of data by sub-splitting the driven path and processing the scan points for each segment separately as a depth image. In the depth domain, we identify foreground objects, such as trees and building facades in the background, by histogram analysis. Subsequently, we remove erroneous scan points and foreground objects, and we fill holes by adaptive interpolation. By triangulation, the processed points are transformed into a geometric facade mesh. Since the camera is calibrated and the image acquisition is synchronized with the laser scanners, we directly project the triangular mesh onto the images, identify for each triangle the corresponding image location, and texture-map the mesh while handling occlusions from foreground objects. Then the facade mesh is optimized for interactive rendering: we create an atlas as an efficient texture representation, generate multiple level-of-details by using mesh simplification algorithms, subdivide the facades models, and combine all submodels in a hierarchical scene graph. This enables us to explore large facade models even with standard VRML viewers.

Finally, we use airborne laser scans to complete the facade models with rooftops and terrain shape. We create a DSM from the scans, convert it to a triangular surface mesh, and texture-map this mesh with aerial images. Due to our specific global localization method, the facade models are automatically registered with the DSM; thus, the only remaining problem to solve is merging surface mesh and facade models in a consistent way. To do so, we mark the location of facades and foreground objects in the map and remove surface mesh triangles corresponding to marked locations. Then we put the facade models in place and create a blend mesh to connect both meshes seamlessly. The result is a texture-mapped 3D model suitable for walk-, drive-, and fly-thrus.

We have applied the developed methods on a large data set of downtown Berkeley and have analyzed both the quality of the results and the processing speed of the algorithms. We have found that our approach is capable of generating a highly detailed, photo-realistic 3D model of the architectural structures. Furthermore, both data acquisition and automated model generation are extremely fast: for five downtown Berkeley blocks, the acquisition time has been only eleven minutes and the total processing time only two-and-a-half hours, thereby greatly outperforming, to the best of our knowledge, all existing methods. Since the complexity of our approach is linear in area size, it is scalable and applicable to large environments.

There are several ways in which this work could be extended in the future. However, the following avenues appear particularly important or promising to us:

Modeling foreground objects
In our current approach, foreground objects, such as trees, cars, traffic signs, light posts and telephone masts, are simply removed, as they are not part of the architectural

structure. However, these objects could substantially contribute in making the street scenery lively and real. Given the complexity and clutter of the foreground scenery, the proper reconstructing of the individual objects appears to us a challenging problem, especially since only one side of the objects is captured, and the scan resolution is not adequate for fine objects, such as a cable mast. A potentially suitable solution could be to identify certain object types, such as trees or cars, and to replace them by generics of comparable size and appearance. Both laser scans and images could be used for this classification, requiring complex computer vision and image understanding techniques.

Airborne model generation
We have shown that it is possible to merge the ground-based facades directly with a surface mesh from an airborne view, with the advantage of not having to make any assumptions about building shapes and at the cost of, at times, jittery edges for areas with no available ground-based facade. The airborne model generation could be greatly extended by using more sophisticated approaches already known in the literature: for example, by combining edge information from both the DSM and the aerial image. Fusion of the thus obtained polygonal model with the ground-based facade models may open up new problems and vistas. Additionally, the planar facades even of extremely tall buildings or backsides in this airborne model could be texture-mapped by using far-away views occasionally visible in the ground-based images. Both could possibly result in higher model quality for the non-ground-based model parts.

Texture reduction and synthesis
Since architectural structures are highly repetitious, it is often possible to find identical texture in many different locations on a facade. This could be exploited for two purposes: first, the amount of necessary texture memory could be further reduced by determining and re-using redundant texture patches. This is particularly important since texture consumes the most memory on the graphics card. Second, if there are extremely large foreground objects, it can occur that some facade triangles are not visible in any camera image and can therefore not be texture-mapped at all during the model generation process. The redundancy in architectural structure could be used to fill these texture holes in a copy-and-paste-like fashion, and we have already obtained encouraging results in preliminary attempts to do so.

References

[Ameri and Fritsch, 2000] B. Ameri and D. Fritsch: "Automatic 3D building reconstruction using plane roof structures", ASPRS Congress 2000, Washington, DC, 2000

[Antone and Teller, 2000] M.E. Antone and S. Teller: "Automatic recovery of relative camera rotations for urban scenes. " Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Hilton Head Island, 2000, p.282-9

[Araujo et al., 1998] H. Araujo, R. L. Carceroni, and C. M. Brown, "A Fully Projective Formulation to Improve the Accuracy of Lowe's Pose-Estimation Algorithm", Computer Vision and Image Understanding, Vol. 70, No. 2, pp. 227-238, May 1998

[Baillard and Zisserman, 1999] C. Baillard and A. Zisserman: "Automatic reconstruction of piecewise planar models from multiple views", Proceedings of the Conference on Computer Vision and Pattern Recognition, 1999, pp. 559-565

[Besl and McKay, 1992] P. J. Besl and N. D. McKay: "A method for registration of 3-D shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, 1992.

[Borenstein et al., 1996] J. Borenstein, B. Everett, and L. Feng: "Navigating Mobile Robots: Systems and Techniques", A. K. Peters, Ltd., Wellesley, MA, 1996.

[Bortwick and Durrant-Whyte, 1994] S. Bortwick and H. Durrant-Whyte: "Dynamic Localization of Autonomous Guided Vehicles", Proc. of Int. Conf. On Multisensor Fusion and Int. Systems, Las Vegas, 1994, p. 92-97

[Brenner et al., 2001] C. Brenner, N. Haala, and D. Fritsch: "Towards fully automated 3D city model generation", Automatic Extraction of Man-Made Objects from Aerial and Space Images III, 2001

[Bukowski and Séquin, 1995] R. Bukowski and C. H. Séquin: "Object Associations: A Simple and Practical Approach to Virtual 3D Manipulation", Symposium on Interactive 3D Graphics, Monterey, pp 131-138, 1995

[Chan et al., 1998] R. Chan, W. Jepson, and S. Friedman: "Urban Simulation: An Innovative Tool for Interactive Planning and Consensus Building", Proceedings of the 1998 American Planning Association National Conference, Boston, MA pages 43-50, 1998

[Chang and Zakhor, 1999] N.L. Chang and A. Zakhor: "A Multivalued Representation for View Synthesis", Proc. Int'l Conference on Image Processing, Kobe, Japan, 1999, vol. 2, pp. 505-509

[Chatila and Laumond, 1985] R. Chatila and J.-P. Laumond: "Position referencing and consistent world modeling for mobile robots", Proceedings of the 1985 IEEE International Conference on Robotics and Automation, 1985.

[Cipolla et al., 1999] R. Cipolla, D. Robertson, and E. Boyer: "PhotoBuilder - 3D Models of Architectural Scenes from Uncalibrated Images", ICMCS, 1999, Vol. 1, pp 25-31

[Cox and Wilfong, 1990] I.J. Cox and G.T. Wilfong, editors: "Autonomous Robot Vehicles", Springer Verlag, 1990.

[Cox, 1991] I.J. Cox: "Blanche - An experiment in guidance and navigation of an autonomous robot vehicle", IEEE Transactions on Robotics and Automation, vol 7, pp 193-204, 1991

[Curless and Levoy, 1996] B. Curless and M. Levoy: "A volumetric method for building complex models from range images", SIGGRAPH, New Orleans, 1996, p. 303-312

[Davis et al., 2002] J. Davis, S. Marschner, M. Garr, and M. Levoy: "Filling holes in complex surfaces using volumetric diffusion", First International Symposium on 3D Data Processing, Visualization, Transmission (3DPVT), Padua, Italy, 2002

[Debevec et al., 1996] P. E. Debevec, C. J. Taylor, and J. Malik: "Modeling and Rendering Architecture from Photographs", Proc. of ACM SIGGRAPH 1996

[Dellaert et al., 1999] F. Dellaert, W. Burgard, D. Fox, and S. Thrun: "Using the condensation algorithm for robust, vision-based mobile robot localization", Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1999

[Dempster et al., 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin: "Maximum likelihood from incomplete data via the EM algorithm", J. Roy. Stat. Soc. (series B) 39, 1-38, 1977

[Dick et al., 2001] A. Dick, P. Torr, S. Ruffle, and R. Cipolla: "Combining Single View Recognition and Multiple View Stereo for Architectural Scenes", International Conference on Computer Vision, Vancouver, Canada, 2001, p. 268-74

[Dorai et al., 1998] C. Dorai, G. Wang, A.K. Jain and C. Mercer: "From Images to Models: Automatic 3D Object Model Construction from Multiple Views", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, No. 1, pp. 82-89, 1998

[Efros and Freeman, 2001] A. Efros and W. Freeman: "Image Quilting for Texture Synthesis and Transfer", Proceedings of ACM SIGGRAPH '01, Los Angeles, USA, 2001

[Elfes, 1987] A. Elfes: "Sonar-based real-world mapping and navigation", IEEE Journal of Robotics and Automation, RA-3(3):249–265, June 1987

[El-Hakim et al., 1998] S. El-Hakim, C. Brenner, and G. Roth: "An Approach to Creating Virtual Environments Using Range and Texture", Proc. ISPRS Commission V Symposium, Hakodate, Japan, 1998

[Faugeras et al., 1998] O. Faugeras, L. Robert, S. Laveau, G. Csurka, C. Zeller, C. Gauclin, and I. Zoghlami: "3D Reconstruction of Urban Scenes from Image Sequences", Computer Vision and Image Understanding, Vol.69, No.3, March 292-309, 1998

[Flynn, 2002] J. Flynn: "Motion from Structure: Robust Multi-Image, Multi-Object Pose Estimation", Master's thesis, Spring 2002, U.C. Berkeley

[Förstner, 1999] W. Förstner: "3D-City Models: Automatic and Semiautomatic Acquisition Methods", Photogrammetrische Woche, Stuttgart, 1999

[Fischler and Bolles, 1981] M. A. Fischler and R. C. Bolles: "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography", Communication Association and Computing Machine, 24(6), pp.381-395, 1981

[Fox et al, 1999a] D. Fox, W. Burgard, S. Thrun: "Markov Localization for Mobile Robots in Dynamic Environments", Journal of Artificial Intelligence Research 11, pp. 391-427, 1999

[Fox et al, 1999b] D. Fox, W. Burgard, F. Dellaert, and S. Thrun: "Monte carlo localization: Efficient position estimation for mobile robots", In Proc. of the National Conference on Artificial Intelligence (AAAI), 1999

[Fox et al., 2000] D. Fox, S. Thrun, F. Dellaert, and W. Burgard: "Particle filters for mobile robot localization", In A. Doucet, N. de Freitas, and N. Gordon, edts, Sequential Monte Carlo Methods in Practice. Springer Verlag, New York, 2000

[Frere et al. 1998] D. Frere, J. Vandekerckhove, T. Moons, and L. Van Gool: "Automatic modelling and 3D reconstruction of urban buildings from aerial imagery", IEEE International Geoscience and Remote Sensing Symposium Proceedings, Seattle, 1998, p.2593-6

[Früh et al., 2000] C. Früh, M. v. Ehr, and R. Dillmann: "Aufbereitung von Laserdaten für ein mobiles autonomes 3D-Meßsystem", Autonome Mobile Systeme, Karlsruhe, Germany, 2000, p. 263-270

[Früh et al., 2001] C. Früh, J. Flynn, H. Foroosh, and A. Zakhor: "Fast 3D model generation for urban environments", Workshop on the Convergence of Graphics, Vision, and Video (CGVV'01), Berkeley, USA, March 2001

[Früh and Zakhor, 2001 a] C. Früh and A. Zakhor: "Fast 3D model generation in urban environments", IEEE Conf. on Multisensor Fusion and Integration for Intelligent Systems, Baden-Baden, Germany, 2001, p. 165-170

[Früh and Zakhor, 2001 b] C. Früh and A. Zakhor: "3D model generation of cities using aerial photographs and ground level laser scans", Computer Vision and Pattern Recognition, Hawaii, USA, 2001, p. II-31-8, vol.2. 2

[Früh and Zakhor, 2002] C. Früh and A. Zakhor: "Data Processing Algorithms for Generating Textured 3D Building Facade Meshes From Laser Scans and Camera Images", Proc. Int'l Symposium on 3D Processing, Visualization and Transmission 2002, Padua, Italy, 2002, p. 834 - 847

[Garland and Heckbert, 1997] M. Garland and P. Heckbert: "Surface Simplification Using Quadric Error Metrics", SIGGRAPH '97, Los Angeles, 1997, p. 209-216

[Gordon et al., 1993] N. J. Gordon, D. J. Salmond, and A. F. M. Smith: "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," Proc. Inst. Elect. Eng. F, vol. 140, pp. 107–113, 1993

[Gruen et al., 1995] A. Gruen, O. Kübler, and P. Agouris (Editors): Proceedings of the Workshop Automatic extraction of man-made objects from aerial and space images. Monte Verità/Switzerland, 1995

[Gutmann and Schlegel, 1996] J.-S. Gutmann and C. Schlegel: "Amos: Comparison of scan matching approaches for self-localization in indoor environments", Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots, 1996

[Gutmann and K. Konolige, 1999] J.-S. Gutmann and K. Konolige: "Incremental Mapping of Large Cyclic Environments", International Symposium on Computational Intelligence in Robotics and Automation (CIRA'99), Monterey, 1999

[Haala and Brenner, 1997] N. Haala and C. Brenner: "Generation of 3D city models from airborne laser scanning data", Proc. EARSEL workshop on LIDAR remote sensing on land and sea, Tallin, Esonia, 1997, p.105-112

[Haala and Brenner, 1999] N. Haala and C. Brenner: "Extraction of buildings and trees in urban environments", ISPRS Journal of Photogrammetry and Remote Sensing 54(2-3), 1999, pp. 130–137

[Hähnel et al., 2001] D. Hähnel, W. Burgard, and S. Thrun: "Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot" Fourth European workshop on advanced mobile robots (EUROBOT'01), 2001.

[Heller et. al., 1996] A. Heller, P. Fua, C. Connolly, and J. Sargent: "The Site-Model Construction Component of the RADIUS Testbed System", Proceedings of the DARPA Image Understanding Workshop, Palm Springs, California, 1996, pp. 345-355.

[Huertas et al., 1999] A. Huertas, R. Nevatia, and D. Landgrebe: "Use of hyperspectral data with intensity images for automatic building modeling", Proc. of the Second International Conference on Information Fusion, Sunnyvale, 1999, p.680-7 vol.2. 2

[Jensfelt and Kristensen, 1999] P. Jensfelt and S. Kristensen: "Active global localization for a mobile robot using multiple hypothesis tracking", Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation, pages 13–22, Stockholm, Sweden, 1999. IJCAI.

[Jiang and Neumann, 2001] B. Jiang and U. Neumann: **"**Extendible Tracking by Line Auto-Calibration", Proc. International Symposium on Augmented Reality, pp.97-103, New York, 2001.

[Kalman, 1960] R.E. Kalman: "A new approach to linear ltering and prediction problems", Trans. of the ASME, Journal of basic engineering, March 1960.

[Kanade and Morita, 1997] T. Kanade and T. Morita, "Three-Dimensional Shape and Motion Recovery from Image Streams," Institute of Electronics, Information, and Communication of Japan, Vol. 80, No. 5, 1997, pp. 479-487.

[Kawasaki et al., 1999] H. Kawasaki, T. Yatabe, K. Ikeuchi, and M. Sakauchi: **"**Automatic modeling of a 3D city map from real-world video", Proceedings ACM Multimedia 1999, Orlando, USA, 1999, p. 11-18

[Kim et al, 2001] Z. Kim, A. Huertas, and R. Nevatia: "Automatic description of Buildings with complex rooftops from multiple images", Computer Vision and Pattern Recognition, Kauai, 2001, p. 272-279

[Koenig and Simmons, 1998] S. Koenig and R. Simmons: "A robot navigation architecture based on partially observable Markov decision process models", In Kortenkamp et al., 1998

[Leonard and Durrant-Whyte, 1991] J.J. Leonard and H.F. Durrant-Whyte: "Mobile robot localization by tracking geometric beacons", IEEE Transactions on Robotics and Automation, 1991

[Leonard and Durrant-Whyte, 1992] J.J. Leonard and H.F. Durrant-Whyte: "Directed sonar sensing for mobile robot navigation", Kluwer Academic Publishers, Boston, 1992

[Lowe, 1991] D. G. Lowe, "Fitting parameterized three-dimensional models to images", Trans. On pattern analysis and machine intelligence, vol. 13, No. 5, 1991, pp. 441-450

[Lu and Milios, 1994] F. Lu and E. Milios: "Robot pose estimation in unknown environments by matching 2D range scans", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1994

[Lu and Milios, 1997a] F. Lu and E. Milios: "Globally consistent range scan alignment for environment mapping", Autonomous Robots, Vol. 4, p. 333-349, 1997.

[Lu and Milios, 1997b] F. Lu and E. Milios: "Robot pose estimation in unknown environments by matching 2D range scans", Journal of Intelligent and Robotic Systems, 18, 1997

[Maas, 2001] H.-G. Maas: "The suitability of airborne laser scanner data for automatic 3D object reconstruction", 3rd Int'l Workshop on Automatic Extraction of Man-Made Objects, Ascona, Switzerland, 2001

[Maas and Vosselman, 1999] H.-G. Maas and G. Vosselman, "Two algorithms for extracting building models from raw laser altimetry datas", ISPRS Journal of Photogrammetry and Remote Sensing 54, 1999, pp. 153–163

[McAllister et al., 1999] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra, and C. McCue: "Real-Time Rendering of Real World Environments", Rendering Techniques '99, Proceedings of the Eurographics Workshop on Rendering, Granada, Spain, 1999

[Moravec, 1988] H.P. Moravec: "Sensor fusion in certainty grids for mobile robots", AI Magazine, Summer 1988.

[Neumann and You, 1999] U. Neumann and S. You: " Natural Feature Tracking for Augmented Reality**",** IEEE Transactions on Multimedia, Vol. 1, No. 1, pp. 53-64, 1999

[Pulli et al., 1997] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle: "Robust Meshes from Range Maps", Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, 1997, pp. 205-211

[Ribarsky et al., 2002] W. Ribarsky, C. Shaw, Z. Wartell, and N. Faust: "Building the Visual Earth," to be published, SPIE 16th International Conference on Aerospace/Defense Sensing, Simulation, and Controls, 2002

[Roumeliotis and Bekey, 2000] S.I. Roumeliotis and G.A. Bekey: "Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 2985–2992, San Francisco, CA, 2000

[Rusinkiewicz and Levoy, 2000] S. Rusinkiewicz and M. Levoy: "QSplat: A Multiresolution Point Rendering System for Large Meshes", Proceedings ACM SIGGRAPH, 2000

[Rusinkiewicz and Levoy, 2001] S. Rusinkiewicz and M. Levoy: "Efficient Variants of the ICP Algorithm", Proc. 3DIM 2001

[Russell and Norvig, 1995] S. J. Russell and P. Norvig: "Articial Intelligence: A Modern Approach", Chapter 17, Series in Articial Intelligence, Prentice Hall, 1995

[Schiele and Crowley, 1994] B. Schiele and J.L. Crowley: "A comparison of position estimation techniques using occupancy grids", Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 1994

[Schroeder et al., 1992] W. Schroeder, J. Zarge, and W. Lorensen: "Decimation of Triangle Meshes", Computer Graphics, Volume 25, No. 3, (Proc. SIGGRAPH `92), 1992.

[Séquin et al, 1993] C. H. Séquin, T. A. Funkhouser, and S. J. Teller: "Interactive Exploration of Building Models," MICRO Project Reports, University of California, pp 126-129, Sept. 1993

[Sequira et al., 1996] V. Sequeira, J.G.M. Goncalves, and M.I. Ribeiro: "3D reconstruction of indoor environments", Proc.. Int. Conf. on Image Processing, Lausanne, 1996, p.405-8 vol.2. 3

[Simmons and Koenig, 1995] R. Simmons and S. Koenig: "Probabilistic robot navigation in partially observable environments**",** Proc. of International Joint Conference on Artificial Intelligence, Montreal, 1995. p.1080-7 vol.2

[Stamos and Allen, 2002] I. Stamos and P. K. Allen, "Geometry and Texture Recovery of Scenes of Large Scale", Computer Vision and Image Understanding (CVIU), V. 88, N. 2, Nov. 2002, pp. 94-118

[Stulp et al., 2001] F. Stulp, F. Dell'Acqua, and R. B. Fisher: "Reconstruction of surfaces behind occlusions in range images", Proc. 3rd Int. Conf. on 3-D Digital Imaging and Modeling, Montreal, Canada, 2001, p. 232-239

[Teller, 1998] S. Teller: "Towards urban acquisition from geo-located images", 6.th Pacific Conference on Computer Graphics and Applications, Pacific Graphics 1998, pp. 45-51

[Thrun et al., 1998] S. Thrun, D. Fox, and W. Burgard: "A probabilistic approach to concurrent mapping and localization for mobile robots", Machine Learning, 1998, (also in Autonomous Robots 5, pp. 253, joint issue)

[Thrun, 2000] Thrun, S: **"**Probabilistic algorithms in robotics", AI Magazine, vol.21, American Assoc. Artificial Intelligence, Winter 2000, p. 93-109

[Thrun et al., 2000] S. Thrun, W. Burgard, and D. Fox: "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping"**,** Proc. of International Conference on Robotics and Automation, San Francisco, 2000, p..321-8, vol. 1. 4

[Thrun et al., 2001] S. Thrun, D. Fox, W. Burgard, and F. Dellaert: "Robust Monte Carlo Localization for Mobile Robots", Artificial Intelligence Journal, 2001

[Triggs et al., 2000] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon: "Bundle adjustment - a modern synthesis", Proc. International Workshop on Vision Algorithms, Corfu, 2000. p.298-372.

[Tsai, 1987] R. Y. Tsai: "A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, pages 323-344

[Turk and Levoy, 1994] G. Turk and M. Levoy, "Zippered Polygon Meshes from Range Images", SIGGRAPH '94, Orlando, Florida, 1994, pp. 311-318.

[Vestri and Devernay, 2001] C. Vestri and F. Devernay: "Using Robust Methods for Automatic extraction of buildings", Computer Vision and Pattern Recognition, Hawaii, USA, 2001, p. I-133-8, vol.1. 2

[Vosselman, 1999] G. Vosselman: "Building reconstruction using planar faces in very high density height data", ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery, Munich, 1999

[Vosselman and Dijkman, 2001] G. Vosselman and S. Dijkman: "3D building model reconstruction from point clouds and ground plans", International Archives of Photogrammetry and Remote Sensing, XXXIV-3/W4:37-43, 2001

[Weidner and Förstner, 1995] U. Weidner and W. Förstner: "Towards automatic building extraction from high-resolution digital elevation models", ISPRS Journal of Photogrammetry & Remote Sensing, 50(4), 1995, pp. 38-49

[Werner and Zisserman, 2002] T. Werner and A. Zisserman: "New Techniques for Automated Architecture Reconstruction from Photographs", Proc. 7th European Conference on Computer Vision, Copenhagen, Denmark, 2002

[Whitaker, 1999] R.T. Whitaker: "Indoor scene reconstruction from sets of noisy range images" Second International Conference on 3-D Digital Imaging and Modeling, Ottawa 1999. p.348-57

[Weiss et al., 1994] G. Weiss, C. Wetzler, and E. von Puttkamer: "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans", Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1994

[Yamauchi, 1996] B. Yamauchi: "Mobile robot localization in dynamic environments using dead reckoning and evidence grids", Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 1996

[Yang and Allen, 1998] R. Yang and P.K. Allen, "Registering, Integrating, and Building CAD Models from Range Data", IEEE Int. Conf. on Robotics and Automation, May 18-20, 1998, Leuven, Belgium, pp. 3115-3120

[Yu et al., 2001] Y. Yu, A. Ferencz, J. Malik: "Extracting Objects from Range and Radiance Images", IEEE Transactions on Visualization and Computer Graphics 7(4), pp. 351-364, 2001

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

[Zhao and Shibasaki, 1999] H. Zhao, R. Shibasaki: "A System for Reconstructing Urban 3D Objects using Ground-Based Range and CCD Images", Proc. of International Workshop on Urban Multi-Media/3D Mapping, Tokyo, 1999

[Zhao and Shibasaki, 2001] H. Zhao, R. Shibasaki: "Reconstructing Urban 3D Model using Vehicle-borne Laser Range Scanners", Proc. of the third International Conference on 3D Digital Imaging and Modeling, 2001, Quebec, Canada

[Zisserman et al., 1999] A. Zisserman, A.W. Fitzgibbon, and G. Cross: "VHS to VRML: 3D Graphical Models from Video Sequences", IEEE International Conference on Multimedia and Systems, Florence, 1999