Streaming Surface Reconstruction from Real Time 3D Measurements

Dipl.-Ing. Tim Bodenmüller

Ph.D. Thesis

TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Realzeit-Computersysteme

Streaming Surface Reconstruction from Real Time 3D Measurements

Dipl.-Ing. Tim Bodenmüller

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. (Univ. Tokio) M. Buss
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. G. Färber, i.R.
	2.	Hon.-Prof. Dr.-Ing. G. Hirzinger

Die Dissertation wurde am **10.06.2009** bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am **19.10.2009** angenommen.

Abstract

The thesis *Streaming Surface Reconstruction from Real Time 3D Measurements* presents a robust method for the fast generation of 3D surface models, its verification, and current and potential applications. The method is designed as an *in-the-loop processing* approach, which enables the use in a visual feedback system that assists operators of manual scanner systems. It outperforms recent approaches to streaming surface reconstruction in its ability to process dynamic, unorganized point data from real time streams.

The presented method iteratively generates a dense and homogeneous triangle mesh by inserting sample points from a real time data stream and refining the surface model locally around each new sample point. A spatial data structure ensures a fast access to growing point sets and continuously updated meshes without restrictions to object size or number of sample points. Thus, a user can scan objects without any a priori knowledge concerning the object's size. Further, the generated model can be accessed any time and thus directly visualized to the user.

This method is suitable for unorganized point sets and is not limited to a certain type of scanner, as the measurements enter the surface reconstruction process as a serial stream of 3D points. The method enables the instant processing of real world data generated with scanner systems, requiring additional per-point attributes that further characterize each measurement. Therefore, a general description of manual scanner systems is developed concerning geometric properties, temporal synchronization, and accuracy.

The method is verified by simulated scans of virtual scenes in order to assess processing time and influences of different process parameters. The virtual scenes represent typical real world situations, including e.g. scans of sharp edges or concavities. Further, sensor noise is applied to the data. This way, the robustness of the method is evaluated, at the same time, its limitations are identified. The interrelation of the parameters is discussed and for each, values are optimized.

The method is verified by simulated scans of virtual scenes in order to assess processing time and influences of different process parameters. The virtual scenes represent typical real world situations, including e.g. scans of sharp edges or concavities. Further, sensor noise is applied to the data. This way, the robustness of the method is evaluated, at the same time, its limitations are identified. The interrelation of the parameters is discussed and for each, values are optimized.

Two exemplary applications prove the versatile applicability of the method: the integration of the method into a visual feedback application for the *DLR Multisensory 3D Modeler* and the processing of huge data sets in the context of cultural heritage preservation.

Zusammenfassung

In der Arbeit *Streaming Surface Reconstruction from Real Time 3D Measurements - Schritthaltende Oberflächenrekonstruktion aus 3D Messungen in Echtzeit* wird eine robuste Methode zur schnellen Erstellung von 3D Oberflächenmodellen sowie deren Verifikation und ausgewählte Applikationen behandelt. Die Methode ist als iterativer Prozess aufgebaut und ermöglicht die Realisierung einer schnellen visuellen Rückkopplung für manuelle Scannersysteme, die den Benutzer bei seiner Arbeit unterstützt. Im Gegensatz zu bisherigen Methoden der Oberflächenrekonstruktion ermöglicht der Ansatz eine schritthaltende Verarbeitung von dynamischen und ungeordneten Punktewolken während deren Erfassung mittels eines manuellen Scannersystems.
Mit der präsentierten Methode wird iterativ ein dichtes und homogenes Dreiecksnetz erstellt, indem kontinuierliche Messpunkte aus dem Echtzeit-Datenstrom eingefügt werden und damit das bestehende Netz lokal verfeinert wird. Dabei ermöglicht eine dynamische, räumliche Datenstruktur einen schnellen Zugriff auf die Daten. Der Benutzer eines manuellen Scanners wird in die Lage versetzt, 3D Modelle zu erstellen, ohne vorher die räumliche Ausdehnung der Objekte zu kennen. Das 3D Modell ist jederzeit verfügbar und kann direkt visualisiert werden.
Die Methode setzt keine Ordnung in der gemessenen Punktemenge voraus und ist nicht auf eine bestimmte Art von Scannersystem beschränkt, da die Messungen zunächst in 3D Punkte umgewandelt werden und als solche nacheinander dem Rekonstruktionsprozess zugeführt werden. Die Methode ermöglicht die schritthaltende Verarbeitung realer Messungen von 3D Scannern und benötigt daher für jeden Messpunkt zusätzliche Kenngrößen. In der vorliegenden Arbeit wird eine generalisierte Beschreibung von manuellen Scannersystemen hinsichtlich Messgeometrie, Zeitverhalten und Genauigkeit entwickelt, die eine einheitliche Berechnung der Kenngrößen ermöglicht.
Der Oberflächenrekonstruktionsprozess umfasst zwei Stufen, die *Schätzung der Oberflächennormalen* und die *Generierung des Dreiecksnetzes*. Dabei wird für jeden eingefügten Messpunkt die zugehörige Oberflächennormale geschätzt. Weiterhin wird die Dichte der Punktewolke global begrenzt. Die Generierung des Dreiecksnetzes verwendet diese geschätzten Normalen, um das Dreiecksnetz lokal auf eine 2D Ebene zu projizieren und es dort zu verfeinern. Eine Verifikation der geschätzten Oberflächennormalen vor der Weiterleitung zur Netz-Generierung stellt sicher, dass Fehlschätzungen verworfen werden.
Die Methode wird anhand von simulierten Scans von virtuellen Szenen verifiziert, um den Einfluss der einzelnen Prozessparameter auf das Ergebnis und die Rechenzeit aufzuzeigen. Die gewählten Szenen entsprechen Situationen, wie sie auch mit realen Objekten zustande kommen können, z.B. scharfe Kanten, Ecken oder auch Konkavitäten. Des Weiteren wird Sensorrauschen in den Daten simuliert, um die Robustheit der Methode zu beurteilen und Grenzen aufzuzeigen. Die Wechselbeziehung zwischen den Einflussgrößen wird diskutiert und optimale Parameter erarbeitet.
Zwei exemplarische Anwendungen zeigen die Vielseitigkeit der Methode: Die Integra-

tion in eine Applikation mit dem *DLR Multisensoriellen 3D Modellierer* und visueller Rückkopplung des Prozessfortschritts, sowie die Bearbeitung von sehr großen Datensätzen zur Erhaltung kulturellen Erbes.

Acknowledgment

This Ph.D. Thesis was written during my employment at the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. I received a lot of support while writhing this thesis and I am deeply grateful for that.

Firstly, I would like to thank the Head of the Institute, Prof. Gerd Hirzinger, for giving me the opportunity to work in this Institute and for always encouraging my work. I also like to thank Prof. Georg Färber from the Institute for Real-Time Computer Systems (RCS) at the Technical University of Munich for supervising this Ph.D thesis and for offering invaluable support and advice on many occasions.

The Institute of Robotics and Mechatronics has assembled some really amazing people, and I have had the opportunity to work and become friends with many of them. Many thanks to all my colleagues in the Institute and especially to the people of the 3D sensing and modeling group - it was a lot of fun working together.

I owe a lot of thanks to Michael Suppa for always supporting me and urging me in my work - even if I got lost in details and other projects. Special thanks also to Christian Rink for extensive discussions, critical opinions, and for continuous proofreading - especially for his valuable suggestions on the right notations. Further, I would like to thank Rainer Konietschke and Mareike Döpke for a fruitfully discussions and for extensive proofreading.

I would also like to thank the members of the service team and the IT group of the Institute - they always helped to overcome any bureaucratic and technical obstacle.

Finally, I would like to than my friends and my family for encouraging me during the last years. Last but not least, thanks to my wife Sylvia for unceasing support, trust, and love.

Munich, June 2009 Tim Bodenmüller

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 1
- 1.2 Contribution of the Thesis . . . 3
- 1.3 Related Work . . . 5
- 1.4 Outline of the Thesis . . . 8

2 Analysis of Manual Scanner Systems **9**
- 2.1 Introduction . . . 9
- 2.2 Range Sensor Description . . . 12
- 2.3 View Alignment and Synchronization . . . 18
- 2.4 Measurement Errors and Accuracy . . . 22
- 2.5 Summary and Discussion . . . 27

3 Streaming Surface Reconstruction **29**
- 3.1 Geometric Definitions . . . 30
- 3.2 Density Limitation . . . 33
- 3.3 Estimation of Surface Normals . . . 36
- 3.4 Selection and Tracking . . . 41
- 3.5 Localized Triangulation . . . 44
- 3.6 Summary and Discussion . . . 49

4 Spatial Data Structures **51**
- 4.1 Dynamic Space Partitioning . . . 51
- 4.2 Application to Streaming Modeling . . . 58
- 4.3 Summary and Discussion . . . 64

5 Verification of Method **65**
- 5.1 Simulations . . . 65
- 5.2 Analysis and Discussion . . . 72
- 5.3 Summary . . . 81

6 Manual Digitization **83**
- 6.1 The DLR Multisensory 3D Modeler System . . . 83
- 6.2 Visual Feedback for Manual Scanning . . . 87
- 6.3 Results . . . 91

7 Large Object Modeling **97**
- 7.1 Data Acquisition and Preprocessing . . . 98
- 7.2 Surface Reconstruction and Post-Processing . . . 101
- 7.3 Results . . . 102

8 Conclusion **107**
8.1 Conclusion 107
8.2 Future work 108

A Computational Geometry **111**
A.1 Distance Metrics 111
A.2 Description of general rotations in 3D space 112
A.3 Intersection of Edges 113

B Calculation of Reference Sample Density **115**
B.1 Helper Equations 115
B.2 Cartesian Geometry 116
B.3 Perspective Geometry 116
B.4 Cylindrical Geometry 117
B.5 Spherical Geometry 118

Bibliography **119**

List of Figures

1.1 Automatic and manual scanning with 3D modeling 2
1.2 Streaming surface reconstruction processing pipeline 4

2.1 Principal components of a manual scanner system 10
2.2 Relation between image coordinates and Cartesian space 13
2.3 Relation between the reference sample densities 16
2.4 Sample density on a tilted reference plane 17
2.5 Comparison of different sample densities for a rotatory DoF 17
2.6 Concept of sensor synchronization and data labeling 20
2.7 Concept of view alignment and serialization 22
2.8 Difference between real and measured sample point 25
2.9 Measurement of the ray error e^* on a known reference plane 26

3.1 The *RT-SSR* process 29
3.2 Edge data structure in a triangle mesh 32
3.3 Gaps between points in a limited point set 34
3.4 Normal estimation at sharp edges 39
3.5 Calculation of the minimum neighborhood radius 40
3.6 Flipped surface normals due to noise and flat line-of-sight 42
3.7 Violations of the variance criteria 43
3.8 Local update of triangulation 47
3.9 The signed tetrahedron volume. 49

4.1 Definition of a voxel 53
4.2 Binary search tree representation of a voxel space 55
4.3 Basic concept of an octree 55
4.4 Access to the children of an octree voxel 56
4.5 Construction of the Extendable Octree 57
4.6 Neighborhood query with linked neighboring voxels 60
4.7 Complexity of octree traversal 61
4.8 Relation between query of voxels and edge neighborhood 62

5.1 Distance-dependent polynomial deviation 66
5.2 Virtual scenes 67
5.3 Generated scans 68
5.4 Progress of mesh generation for Scan *S1E-N* 70
5.5 Surface Reconstruction results 71
5.6 Density limitation with different radii 72
5.7 Comparison of simple limitation and replacement limitation 73
5.8 Replacement limitation for two scans with similar noise 74
5.9 Influence of $R_{\min}$ and k_n on the normal estimation 75
5.10 Rejection of points due to a too small value of k 75

5.11 Rejection of incorrect surface normals 76
5.12 Flipped surface normals caused by a flat line-of-sight 76
5.13 Interaction between regular selection and fast selection 77
5.14 Coupling between $R_{e_{\min}}$ and $R_{e_{\max}}$ during mesh generation 78
5.15 Computational effort for neighborhood queries 80
5.16 Voxelization on Scan *S2-N* 80

6.1 The DLR Multisensory 3D Modeler 84
6.2 Function of the 3DMo-LSP sensor 86
6.3 Precisions of the 3DMo range sensors 87
6.4 Concept of the 3DMo visual feedback system 90
6.5 Mesh generation and augmented visualization of the progress 91
6.6 Generated 3D model of a putto statue 92
6.7 Scans and 3D models of two busts 93
6.8 Photorealistic 3D models for different applications 94
6.9 Registration of scanned faces with MRI data 95
6.10 Inspection of the registration for a wooden workpiece 96

7.1 *Z+F Imager 5003* and *Panoramic Camera* 98
7.2 Automatic filtering of range images 100
7.3 Modeled parts of Neuschwanstein Castle 103
7.4 Model of the king's room in the gatehouse 104
7.5 Texturing of the models 104
7.6 Throne room of Neuschwanstein Castle 105
7.7 Reconstruction of a roof 106
7.8 Model of a factory building 106

Abbreviations and Symbols

Abbreviations

3DMo	Multisensory 3D Modeller
AABB	Axis-aligned bounding box
CCD	Charge-coupled Devices
CMM	Coordinate Measurement Machine
DoF	Degree of Freedom
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Cener)
GPU	Graphics processing unit
IMU	Inertial Measurement Unit
LRS	Laser Range Sensor
LSP	Light Stripe Profiler
MLS	Moving least squares
RT-SSR	Streaming Surface Reconstruction from Real Time Data
SCS	Stereo Camera Sensor
SGM	Semi Global Matching
SSR	Streaming Surface Reconstruction
ToF	Time-of-Flight

Mathematical Notation

a	Scalar value
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$\mathbf{a} \times \mathbf{b}$	Cross product
$\langle \mathbf{a}, \mathbf{b} \rangle$	Dot product
$\|\mathbf{a}\|$	Vector length or euclidean norm
$\mathcal{X} = \{x_1, \ldots, x_n\}$	Set with n elements

List of Symbols

Description of Scanner Systems

d_{min}, d_{max}	Minimum and maximum values of distance measurements
$u(i), v(j)$	Grid coordinates of a distance pixel
u_0, v_0	Offsets of grid coordinates
$\Delta u, \Delta v$	Sample widths of grid coordinates
d_{ij}	Distance measurement at i-th row and j-th column
$\mathbf{D}$	Range image i.e. matrix of all distance pixel
$\mathbf{p}_{ij}$	Sample point corresponding to distance measurement d_{ij}
$\mathcal{P}_{\mathbf{D}}$	Set of all sample points from the range image $\mathbf{D}$
${}^{w}\mathbf{T}_{s}$	Transformation from sensor coordinates to world coordinates
${}^{l}\mathbf{T}_{s}$	Transformation from sensor coordinates to local coordinates (pose measurement)
${}^{w}\mathbf{T}_{l}$	Transformation from local coordinates to world coordinates (calibration matrix)
$\mathbf{p}$	A sample point
$\mathbf{s}$	Line-of-sight or ray direction
$\mathbf{o}$	Ray origin
d_{ray}	Ray length
δ	Reference sample density
σ	Expected deviation

Parameters of the *RT-SSR* method

$R_{\min}$	Limitation radius of density limitation
$R_{n0}, R_{n_{user}}$	Initial neighborhood radius
$R_{n_{user}}$	User override for the initial neighborhood radius
k_n	Maximum number of neighboring points
$\alpha_{s_{\max}}$	Maximum valid grazing angle
$\alpha_{n_{\max}}$	Maximum fast selection angle
$n_{s_{\min}}$	Minimum number of selected neighbors
$R_{e_{\min}}$	Minimum edge length and limitation radius
$R_{e_{\max}}$	Maximum edge length and neighborhood radius

Estimation of Surface Normals and Selection

$\mathbf{p}, \mathbf{q}$	Sample points

$\mathbf{n}$	Surface normal
$\mathcal{P}$	Set of sample points in the normal estimation
$\mathcal{N}_R(\mathbf{q}, \mathcal{P})$	Point ball neighborhood of $\mathbf{q}$ in $\mathcal{P}$
$\mathcal{N}_{R,k}(\mathbf{q}, \mathcal{P})$	k-in-R point neighborhood of $\mathbf{q}$ in $\mathcal{P}$
$\mathcal{P}_\mathcal{N}$	Set of neighboring points for normal estimation
$\mathbf{Cov}$	Covariance matrix of point neighborhood
$\mathbf{c}$	Mean (center) of point neighborhood
$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$	Eigenvectors of $\mathbf{Cov}$
$\lambda_1, \lambda_2, \lambda_3$	Eigenvalues of $\mathbf{Cov}$
$\mathfrak{c}_0, \mathfrak{c}_1, \mathfrak{c}_2, \mathfrak{c}_3$	Selection criteria
$\mathfrak{c}_{fast}$	Fast selection criterion
$\sigma_1^2, \sigma_2^2, \sigma_3^2$	Variances of normal estimation
$\alpha_{s_{\max}}$	Maximum valid grazing angle
$\alpha_{n_{\max}}$	Minimum required number of selected points for fast selection
$n_{s_{\min}}$	Maximum directional difference for fast selection
$\beta_{\min}$	Minimum change in normal direction for tracking

Mesh Generation

$\mathbf{v}$	Vertex of a mesh
$e = \overline{\mathbf{ab}}$	Edge connecting two vertices $\mathbf{a}$ and $\mathbf{b}$ in a mesh
$t = \Delta(\mathbf{a}, \mathbf{b}, \mathbf{c})$	Triangular face connecting three vertices $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ in a mesh
$\mathcal{M}$	Mesh
$\mathcal{V}_\mathcal{M}$	Set of vertices of $\mathcal{M}$
$\mathcal{E}_\mathcal{M}$	Set of edges of $\mathcal{M}$
$\mathcal{T}_\mathcal{M}$	Set of triangles of $\mathcal{M}$
$\mathcal{E}(\mathbf{v})$	Edges that are connected to the vertex $\mathbf{v}$
$\mathcal{N}_R^\mathcal{V}(\mathbf{q}, \mathcal{M})$	Vertex ball neighborhood of a vertex $\mathbf{q}$ in the mesh $\mathcal{M}$
$\mathcal{N}_R^\mathcal{E}(\mathbf{q}, \mathcal{M})$	Edge ball neighborhood of a vertex $\mathbf{q}$ in the mesh $\mathcal{M}$
$\mathcal{E}_R(\mathbf{q}, \mathcal{M})$	Set of edges in the edge ball neighborhood $\mathcal{N}_R^\mathcal{E}(\mathbf{q}, \mathcal{M})$
$\mathcal{V}(\mathcal{E}_R(\mathbf{q}, \mathcal{M}))$	Set of vertices in the edge ball neighborhood $\mathcal{N}_R^\mathcal{E}(\mathbf{q}, \mathcal{M})$
$\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}})$	Projection of $\tilde{\mathbf{v}}$ onto the tangent plane of $\mathbf{v}, \mathbf{n}$
$\mathcal{C}(\mathbf{v})$	Set of candidate vertices for triangulation
$\mathcal{L}(\mathbf{v})$	Set of neighboring edges on the tangent plane of $\mathbf{v}$

$\mathcal{E}_\mathcal{C}(\mathbf{v})$	Set of candidate edges for triangulation
$\vartheta(\ \mathcal{E}_\mathcal{C}(\mathbf{v})\ )$	Set $\mathcal{E}_\mathcal{C}(\mathbf{v})$ sorted w.r.t. their edge length
$\mathcal{V}_\mathcal{E}(\mathbf{v})$	Set of vertices that are connected to $\mathbf{v}$ by an edge
$\mathcal{C}_\mathcal{T}(e)$	Set of candidate triangle points for triangle face calculation

Spatial Data Structures

$\mathbf{i}$	Grid coordinate of a voxel
l	Voxel size (edge length) of an voxel
k	Level of an octree voxel
l_k	Edge length of an octree voxel at level k
$\mathcal{P}_\mathbf{i}$	Set of points or vertices that are inside of voxel $\mathbf{i}$
$\mathcal{D}_\mathcal{P}$	Set of non-empty voxel
$\mathcal{D}_\mathcal{N}$	Set of voxel that intersect with a point ball neighborhood $\mathcal{N}$
$\mathcal{P}_{\mathcal{D}_\mathcal{N}}$	set of points or vertices inside of $\mathcal{D}_\mathcal{N}$
$\mathcal{D}_{\mathcal{N}\mathcal{E}}$	The set of voxels that intersect with the edge ball neighborhood
$\mathcal{E}_{\mathcal{D}_{\mathcal{N}\mathcal{E}}}$	The set of edges that have at least one end point inside of $\mathcal{D}_{\mathcal{N}\mathcal{E}}$

1 Introduction

In recent years, different 3D scanner systems that allow for fast and precise digitization of real objects have been developed. Many of them are manually operated systems i.e. a human user moves the system along the surface. These systems are more flexible compared to automatic systems, since the operator can freely move the device and thus is able to digitize challenging surfaces that require a complex scan path. However, the user needs a visual feedback in order to monitor the digitization progress.
In this thesis, a novel streaming 3D surface reconstruction method for visual feedback that generates a triangular mesh directly from a 3D point stream measured by a manual scanner systems in real time is presented. The proposed algorithm features the fast and dynamic generation of a triangular mesh directly from a stream. Neither a priori knowledge about the size or shape of the digitized scene is assumed nor a spatial order of the incoming points is required.
In the following sections, the motivation for streaming 3D surface reconstruction in the context of manual scanning and its major challenges are described. Further, the contribution of this thesis is presented and previous work is summarized.

1.1 Problem Statement

The rapid generation of 3D models of arbitrary objects and environments is highly requested in various fields. Applications range from workpiece inspection and rapid prototyping in automation to virtual catalogs for e-business or the design of scenes for games and film industry. In the domain of robotics and automation, fast and dynamic generation of 3D models for work cell exploration, collision avoidance, and grasp planing are needed.
Various commercial and research 3D scanner systems that meet these requirements have been developed in recent years. Systems can be categorized into volumetric scanners such as CT[1] or MRT[2] systems and surface scanners that sample the surface of an object. The latter are subject of this work. Surface scanner systems range

[1] CT=computer tomography
[2] MRT=magnetic resonance tomography

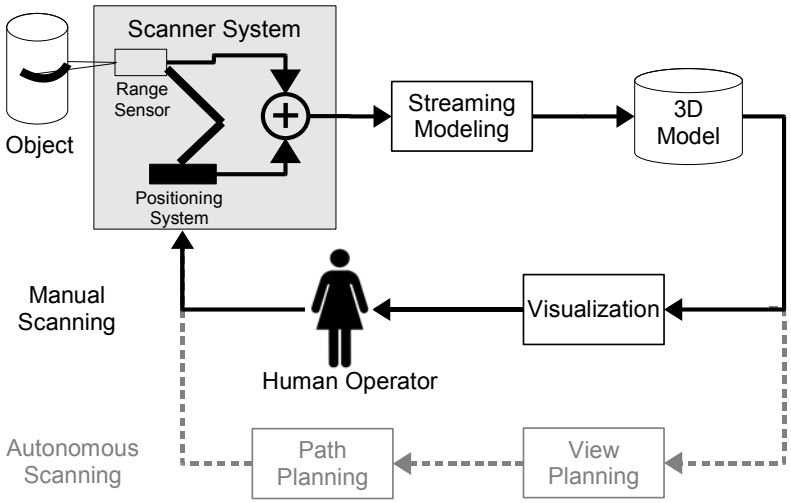


Figure 1.1: Automatic and manual scanning with 3D modeling: The range sensor is mounted onto a positioning device (gray box). The views of the range sensor are merged with the pose data to generate globally aligned 3D points. The 3D modeling process is fed with this 3D point stream and updates the 3D model. In the case of manual scanning (black path), the 3D model is transfered into a real time visualization. The operator uses the rendered model for view planning and moving the device. In case of automatic scanning (gray path), the view and path planning is performed autonomously.

from very high precision systems for digitizing small objects up to large laser radars for the acquisition of whole buildings. The systems differ e.g. in working range, size, precision, and measurement rate. Depending on the application, different attributes of the sensors are important. As an example, for rapid prototyping and reverse engineering, precision is the dominant criterion, while in the field of robotics, size, weight, and measurement rate outrank the precision. The core component of every surface scanner is a range sensor that measures an ordered set of distance values w.r.t. its native sensor coordinate system. However, multiple measurements from different view points are needed to sample a surface completely. Hence, the sensor has to be moved relative to the scanned surface, either manually or autonomously.

Simple automatic systems either move the scanner system or the scanned object along a single DoF, examples are turn-table systems or body scanners with a linear axis. These systems usually have a very restricted scan path and a limited workspace, enabling 3D modeling by simple and fast volumetric reconstruction methods that utilize the limitations of the device concerning the workspace. If the shape of an object is more complex, these simple scanning systems often fail to sample the complete surface e.g. due to occlusions, holes, or local concavities. Autonomous systems with many DoF, e.g. industrial robots, can overcome these restrictions. However, a complex view and path planning has to be performed to guarantee a complete coverage of the surface. This approach requires a loop closure from the distance measurements to the pose commanding, i.e. an instant integration of new measurements into a 3D model and the planning of new views based on this model. This concept, also known as autonomous exploration and inspection, as described e.g. by Suppa [SKL+07], is illustrated in Fig. 1.1 (gray dotted path).

Alternatively, manually operated scanner systems can be used to digitize complex objects, e.g. hand-guided systems or other manually commanded devices. Here, complicated planning stages are not needed, as the systems benefit from the human operator that performs the view and path planning. The user has to keep track of

the sufficiently scanned surface parts and those that remain to be sampled. Consequently, a real time visualization that supports the operator at this planning task is required. The loop from the measurement to a scanner movement through the operator is closed by visual feedback. In Fig. 1.1 this closed system is depicted (black path) in comparison to an autonomous system.
Providing a suitable visual feedback to the user requires an in-the-loop integration and processing of range measurements. The rendering of the raw measurement data is the most simple way of providing visual feedback, however, it results typically in a poor visualization, as no reasonable shading of the data is possible and it is difficult for a user to judge the quality of a potential 3D model that is generated afterwards. Hence, it would be beneficial for the user to generate the desired 3D model in-the-loop and to visualize it in real time.
An in-the-loop 3D modeling directly from a manual scanner system's real time stream has several challenges: First, no a priori knowledge concerning the shape and the size of the scanned objects is available and no spatial ordering of the incoming stream data can be assumed. In detail, two consecutive measurements can be acquired from different views or can sample a completely different part of the scene. Newly measured data has to be integrated dynamically and rapidly by extending and refining the existing 3D model incrementally. This requires data structures that integrate new data dynamically and enable fast access to the stored data. Due to the manual movement of the device, the sampling of the surface is not necessarily homogeneous or at least sufficiently dense in all regions. Especially hand-held scanner systems that allow for a full 6 DoF motion are difficult to direct at a constant speed for a human operator and thus a uniform movement of the device can not be guaranteed. Here the challenges are twofold: On the one hand, undersampled regions and mismeasurements have to be detected. The integration of such data into the 3D model has to be delayed until enough sample points are available. On the other hand, at least a coarse model for visualization has to be generated as rapidly as possible.

1.2 Contribution of the Thesis

In this work, the in-the-loop generation of a 3D model for visual feedback in the context of manual scanning is tackled by a **streaming surface reconstruction (SSR)** approach. A dense and homogeneous triangular mesh is generated incrementally from a real time stream of 3D measurements, denoted as **real time streaming surface reconstruction (RT-SSR)**. The method takes scanner characteristics into account, but is not limited to a certain scanner system. Moreover, it features a spatial data structure that allows for fast access to growing point sets and changing meshes without restriction in object size or number of sample points. A user can instantly begin scanning objects and does not need to parametrize the work space before. Further, the generated model is available at any time and no additional post-processing is required for visualization.
The proposed surface reconstruction incrementally generates a homogeneous triangular mesh by extending and refining the surface model with every newly inserted point. A generalized description of scanner systems, concerning geometry, spatial and temporal alignment of views, and accuracy is developed and used to derive generalized per-point attributes that are attached to each point. These additional attributes are used during reconstruction to control the process parameters locally w.r.t. to the sensor properties but without the loss of generality. The processing implicitly filters

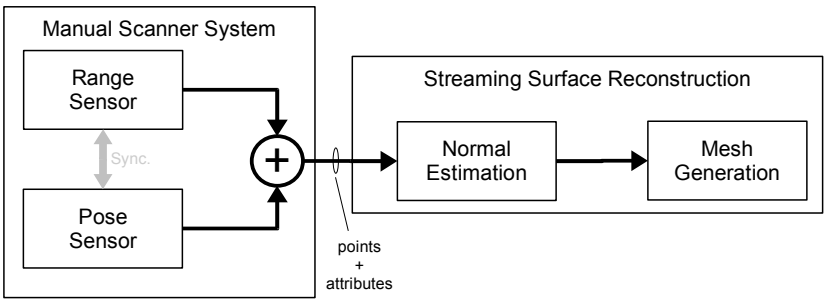


Figure 1.2: Streaming surface reconstruction processing pipeline: The surface reconstruction is fed by a stream of 3D points from a manual scanner system. The system is defined to consist of a range sensor and a pose sensor. The data of both sensors is synchronized and merged to 3D points. The surface reconstruction consists of two principal stages, the normal estimation and the mesh generation.

outliers and rejects undersampled regions until enough sample points are available. The surface reconstruction is divided into two principal steps: the normal estimation stage and the mesh generation stage. In Fig. 1.2, the principal processing pipeline with the embedded reconstruction steps is illustrated. Here, the stream of aligned 3D points is generated from a scanner system that consists of a range sensor and a pose sensor. Both are temporally synchronized and provide measurements at a constant rate. The two outgoing data streams are used to generate a single stream of globally aligned 3D points that is fed into the reconstruction stage.

The reconstruction method is supported by a spatial data structure that provides fast access to the stored data, contrary to volumetric surface reconstruction methods that use the discretized volumes for reconstruction. Hence, the resolution of the data structures is low and requires only little memory. The used data structure grows dynamically with the stored data and has no limitations in size or in the number of data elements. Consequently, a scanned object can be assumed as a priori unknown in shape and size. The reconstruction method operates only on local subsets of the sample point set, thus the required global operations on the complete data set are minimized. The computational complexity is kept low but not constant due to the generality concerning object and data size. However, restrictions of the method and the data structure towards a hard real time processing are also discussed.

The *RT-SSR* method is verified with simulated sensor data and the influence process parameters have on the resulting mesh are shown. Moreover, two applications to the streaming surface reconstruction method are presented. The first is the manual scanning of small objects, which is the primary scope of this work. Here, the integration of the visual feedback with the *DLR Multisensory 3D Modeler* scanner system is shown. As second applications, an application beyond manual scanning is presented: The out-of-core modeling capabilities for large objects and huge data sets, using a long-range laser radar are demonstrated.

1.3 Related Work

In past years, the problem of surface reconstruction from unorganized point sets has been in the focus of computer graphics and vision research. A variety of methods for different applications has been published. Most of them are either designed for the generation of a 3D model in a post processing step (e.g. [Hop94], [GKS00], [LCOL07]), or are streaming processing methods in the context of out-of-core processing of huge data sets (e.g. [Paj05], [ACA07]). Classical approaches convert the complete point set into a surface model after acquisition. In terms of signal processing this approach is denoted as **off-line method**. For the processing of huge data sets so-called **out-of-core methods** that incrementally build a model from a (file-) stream of ordered 3D points are used. Consequently, the input points must not be kept in memory completely, only a small part is loaded at once. These *SSR* methods are differ from the *RT-SSR* method applied in this work, since the processed data set is static i.e. the extension and the number of points are known a priori and the amount of data does not grow during processing. Further, the data sets are often assumed to be at least partially spatially ordered, e.g. sorted along an axis. Other methods process the data in-the-loop and are denoted as **on-line method** (e.g. [HI00], [RHHL02], [ZSK06]). These approaches are mostly used for processing measurements of a certain type of acquisition system or to generate a set of disconnected surface patches.
In the following, an overview of existing surface reconstruction methods is given. The methods are divided into general methods for unorganized point sets, and methods that use the local spatial order of the range data and hence are specifically designed to measurements of a certain scanner system.

1.3.1 Methods for Unorganized Point Sets

Some methods focus on the generation of topologically correct and closed surfaces from an unorganized point set that represents a fully sampled object. In the works of Edelbrunner [EM94] and Bajar et al. [BBX95] the so-called α-shapes for creating a close and topologically correct surface from a point set are used. A work of the same category has been published by Attene and Spangnuolo [AS00], which extends the *sculpturing algorithm* of Boissonnat [Boi84] by graph-based constraints. More recent methods are the *Tight Cocone* algorithm published by Dey et al. [DG03] and the *Power Crust* algorithm by Amenta et al. [ACK01]. All of these methods are limited to point sets of closed surfaces. Hence, they are hardly suitable for streaming surface reconstruction, since the scanned surface is typically incomplete: only a part is digitized, and moreover, the sample point set grows dynamically.
An early approach to surface reconstruction of non-closed surfaces with arbitrary topology was presented in the work of Hoppe [Hop94]. This method can be divided in three processing steps: (1) the generation of a dense, homogeneous mesh that coarsely fits the point set, (2) an optimization of the vertex positions and mesh topology, and (3) the generation of a piecewise smooth subdivision surface of the mesh and the input point set. In the context of this thesis the generation of an initial homogeneous, dense mesh is relevant (first published in [HDD$^+$92]). First local surface information is estimated by fitting tangent planes for every point of the input point set, resulting in corresponding surface normals. These are used to define an implicit function and generate a discretized volumetric representation. Finally, a homogeneous mesh is generated, using the marching cubes algorithm from Lorensen and Cline [LC87]. In the work of Wang et al. [WOK05] another volumetric approach that is more robust to

irregular and sparse sample data is presented. Here, first the point set is voxelized and a topological thinning is applied. Finally, a surface is generated by using an extended version of the algorithm of Azernikov et al. [AMF03].
Contrary to the approach of Hoppe, the Localized Delaunay Triangulation method of Gopi [GKS00] does not make use of a voxelization but generates a mesh directly from the point set. For every point a surface normal is estimated first, followed by a local triangulation for all points. During the triangulation, the spatial neighboring points for each point are mapped to the tangent plane of the respective point. So-called candidate points that are potential direct neighbors on the surface, are searched and used to generate a local Delaunay triangulation. Further, Gopi focuses on finding sampling conditions for an optimal neighborhood in the triangulation stage.
A different approach is presented by Alexa et al. [ABCO+03]. Here, a local approximation of the surface with polynomials using the *Moving Least Squares (MLS)* is used to generate a smooth and homogeneous point set that can instantly be used for rendering. Fleishman et al. [FCOS05] present a MLS-based approach based on the work of Levin [Lev03]. The method features a piecewise smooth surface reconstruction from potentially noisy point sets. Lipman et al. [LCOL07] extend the MLS reconstruction by an indicator for local singularities, resulting in a faithful surface reconstruction with scattered data and local discontinuities.
Recent methods also tackle the problem of out-of-core surface reconstruction from huge data sets where the point set is not completely kept in memory but partially loaded from file. Pajarola [Paj05] presents a general framework for stream-based processing of large data sets, using a sweep plane approach, i.e. the point set in the file is assumed to be sorted along the z-axis. Hence, the point set can be processed by sliding a window volume over it, keeping only the points inside the window's volume in memory. Basic stream operations, e.g. normal estimation, curvature estimation, or smoothing, are presented. In the work of Allègre et al. [ACA07] a streaming surface reconstruction algorithm for closed surfaces that bases on the convection technique of Chaine [Cha03] is presented. The input data must be organized into a stack of slices along a coordinate axis.

1.3.2 Approaches using Organized Range Data

The second category of methods is directly related to scanner systems, as they use the local ordering of the measured data for reconstruction of local surfaces. Most approaches are limited to 2D range images (2.5D images), i.e. the measured data is a 2D grid of distance values. Such methods work for data generated by stereo reconstruction or structured light systems but exclude single-stripe systems (e.g. light stripers) or touch probes.
Common approaches of this category first generate local surfaces from each range image and merge the surfaces into a single 3D model afterwards. The first step can be processed on-line, the second step is always performed off-line. In the work of Rusinkiewicz et al. [RHHL02] a manual structured light system is presented that generates 2D range images. The images are registered during the acquisition i.e. in real time, requiring strongly overlapping areas in the images. For visual feedback, the raw point data is visualized using a point splatting algorithm [RL00]. The generation of a single, final mesh is entirely performed as a post-processing step, generating an initial triangulation for each range image and subsequently zippering [TL94] the meshes to a single 3D model.
A similar approach is chosen in the work of Hilton and Illingworth [HI00]: 2.5D range

images are separately meshed in 2D using a step discontinuity triangulation. The initial range image triangulation is performed on-line, providing a visual feedback to the operator. The approach is extended to 1D laser-stripe systems by merging adjacent stripes into a 2D range image. In a post-processing step, a merged 3D model is generated using a volumetric approach.
Zach et al. [ZSK06] combine the above approach with the work of Curless and Levoy [CL96]. Instead of creating an implicit volumetric description, the polygonal representation is used directly. The resulting approach overcomes limitations in memory and can be implemented on a GPU, resulting in rapid processing.
A different approach to the generation of 3D models from multi-view stereo reconstruction using region growing is presented by Habbecke and Kobbelt [HK07]. In this approach the surface is approximated by planar discs. First an initial seed of discs is created. The discs are grown and new discs are added until the approximation is sufficient. This method is primarily designed to overcome problems of sparse range image reconstruction caused by untextured regions. Finally, Fiorin et al. [FCS07] present an MLS-based approach to the reconstruction of a triangle mesh from huge 2.5D range images.

1.4 Outline of the Thesis

In Chapter 2, manual scanner systems are in detail analyzed w.r.t. geometrical properties, system synchronization, and sensor accuracy. A standardized description of mapping into Cartesian space is derived and common per-point attributes are defined. Further, the alignment with a corresponding pose as well as the temporal synchronization between pose and distance measurements are discussed. Finally, the sensor accuracy is analyzed and a per-point quality criterion is defined.
Chapter 3 describes the *RT-SSR* method. Its two principal stages, the normal estimation and the mesh generation, are detailed and the functional steps and process parameters are explained.
In Chapter 4, concepts for spatial data structures and their properties are discussed. Further, the rapid access to dynamic data sets by spatial data structures in the context of the streaming surface reconstruction is detailed.
The verification of the surface reconstruction method applying simulated measurements is presented in Chapter 5. The coupling between the process parameters is analyzed and their influence on reconstruction process and processing time is discussed. In Chapter 6, the design of a visual feedback system with the streaming surface reconstruction for an actual manual scanning system is explained. In the subsequent Chapter 7, an application beyond manual scanning is shown. Here, the reconstruction algorithm is applied for modeling of large objects, using the example of modeling building interiors from data of a long-range laser radar.
Chapter 8 summarizes and concludes the thesis. An outlook to future work is presented.

2 Analysis of Manual Scanner Systems

The concept of a 3D scanner system is used for various types of digitizing systems. A general definition is as follows: 'A 3D scanner is a device that analyzes a real world object or environment to collect data on its shape and possibly color'[1]. However, this definition is too unspecific for the manual scanning applications addressed in this work.
Generally, manual scanning is the task of sampling the surface of an object by manually moving a scanner device along a scan trajectory w.r.t the surface. A **manual scanner system** consists of a **range sensor** that measures a set of distances and a **pose sensor** that measures the pose of the scanner system w.r.t. a global coordinate system, as illustrated in Fig. 1.2.
In this chapter, an overview of recent scanner systems is given and the components of manual scanner systems are analyzed. Further, a generalized geometric description that enables the calculation of globally aligned 3D coordinates and per-point attributes from multiple views is derived. Moreover, the temporal synchronization of sensor data and the matching of range images with their corresponding poses, as well as the accuracy of scanner systems are addressed.

2.1 Introduction

Today, a large variety of range sensors and complete 3D scanning systems exists. The systems are designed for miscellaneous tasks and therefore vary in measurement principle, size, accuracy and resolution. Not all scanner systems are suitable for manual scanning applications. In Fig. 1.1 the concept of manual scanning has been introduced. The presented closed-loop concept of manual scanning implicates that the components generate measurements at a constant rate and can provide the measured data in real time. Hence, temporal synchronization of hardware components and measurement data is necessary. In Fig. 2.1, the principal components of such a synchronized manual scanning system are illustrated.
One core component of a manual scanner system is the **range sensor** i.e. a device measuring the distance to all surfaces in its field of view. Many commercial and re-

[1] From Wikipedia at `http://en.wikipedia.org/wiki/3d_scanner`, 2008

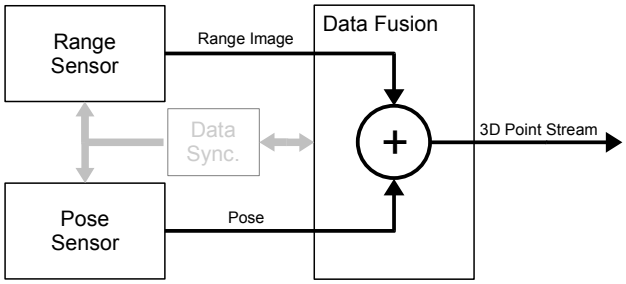


Figure 2.1: Principal components of a manual scanner system: It consists of two core components, a range sensor and a pose sensor, both continuously measuring. The fusion of both measurements results in a stream of globally aligned 3D points. Sensors and data fusion are additionally connected by a data synchronization module.

search range sensors are promoted as scanner systems. However, they do not comply with the definition as used in this work, as a single shot or a single view of a range sensor is generally not sufficient to sample an object's surface completely. Thus, measurements of multiple views have to be aligned into a common space. In the context of manual scanning, this alignment must be performed in real time. Consequently, the second mandatory component of a manual scanner system is a **pose sensor** that measures the location and orientation of the scanning device for each range image.

Range Sensors

Miscellaneous approaches to a contactless measurement of distances exist. In this work, only optical systems are regarded while other techniques are also possible. They range from small, high precision digitizers for rapid prototyping (e.g. the *T-Scan2*[2]) to large laser-radars (e.g. the *Imager 5003*[3]) as used for land surveying and archiving of cultural heritage.
Range sensors can be categorized w.r.t. the physical measurement principle. The most common are triangulation-based systems and Time-of-Flight systems (ToF). A good comparison of different measurement principles is found in the early work of Besl [Bes88]. Triangulation-based systems measure distances by determining the height of a triangle, spanned by two non-parallel rays onto the same sample point. They can be further divided into active and passive triangulation systems. Active triangulation systems use their own light source, i.e. the measurement triangle consists of the ray of the light source on the sampled surface and the reflected ray that is captured by an optical sensor. Examples for this type of systems are structured light sensors that project a 2D light pattern onto the surface, light stripe sensors that use a single laser line, or systems with point-wise measurements. Structured light systems are described in the work of Young et al. [YBD+07], Ishii et al. [IYDT07], or Takei et al. [TKH07]. A commercial system is the *Comet IV system*[4]. Light stripe sensor systems are described by Suppa et al. [SKL+07] and Winkelbach et al. [WMW06]. This is also the most popular commercial system, examples are the *Metris Modelmaker*[5] or the *Faro Laser Line Probe*[6].
In contrast, passive systems consist of two camera sensors and match the reflected light of the global illumination in both images. The corresponding method is de-

[2]Steinbichler T-Scan2, see `http://www.steinbichler.de`, 2009
[3]Zoller+Fröhlich Imager 5003 see `http://www.zf-laser.com`, 2009
[4]Steinbichler Comet IV`http://www.steinbichler.de`, 2008
[5]Metris Modelmaker `http://www.metris.com`, 2009
[6]Faro Laser Line Probe `http://www.faro.com`, 2009

noted as stereo reconstruction, e.g. used in the *Point Grey Bumblebee 2*[7]. A public comparison and benchmark of recent stereo reconstruction algorithms is provided by Scharstein and Szeliski[8]. Recent approaches to triangulation-based range sensing combine the passive stereo reconstruction and active methods, e.g. the systems described by Deng et al. [DNZ06], Davis et al. [DNRR05], or Weise et al. [WLG07].
ToF systems measure the time between sending a beam from a pulsed light source and receiving its reflection. The measured time or phase shift directly corresponds to the distance between the sensor and the object. Commercial systems of this type are e.g. the *Sick LMS 200*[9], the *SR-3000*[10], or the *PMD[vision] 19k*[11].
A range sensor can also be classified w.r.t. its sensor properties. The most important ones for 3D modeling are the type of image coordinate system and geometric mapping, the image resolution, and the sensor accuracy. Usually, the distance measurements of a single view are a 2D ordered set, i.e. for every view a matrix of distance pixels is measured. This matrix is called **range image**. The image coordinate system is the native coordinate system the range image geometrically refers to. The image resolution encodes the size of the range image, i.e. the number of samples per view. Sensors can be divided into two dimensional sensors i.e. sensors measuring a (2D) matrix of distances, one dimensional sensors providing a (1D) stripe of distances, and scalar sensors that acquire a single value per measurement or view. The geometric mapping defines the mapping between the image coordinate system and the corresponding Cartesian coordinate system. For example, the mapping for camera-based systems can be described by a pin hole model or a perspective transformation respectively. The general description of range sensors using sensor geometry and resolution is detailed in Section 2.2. Temporal properties of sensors, e.g. measurement rate or communication delays, are important w.r.t. real time operations and will be discussed in Section 2.3. Another important property is the accuracy of the sensor which is discussed in Section 2.4. Further attributes such as size, weight, or interface are important criteria for the suitability for a specific application. However, they do not influence the measured data and thus are not further examined in this work.

Pose Sensors

The pose of a scanning device can be measured in several ways. Usually, external devices are used, however, algorithms that use the sensory data of the range sensor itself to determine the pose are also possible.
A variety of external devices exists that can be used for pose measurement. This includes optical tracking systems, Coordinate Measuring Machines (CMM), and robotic manipulators. Optical systems are used, if a large working volume is required or a hand-guided system that is not attached to any additional device is requested. Examples are the *ARTtrack*[12] system or the *Polaris tracker*[13]. For applications that require a high accuracy, CMMs are used, e.g. the *Faro Platinum arm*[14] or the *MicroScribe*[15].

[7] Point Grey Bumblebee 2 http://www.ptgrey.com, 2008
[8] The comparison of stereo reconstruction algorithms is found at the vision group of the Middlebury college http://vision.middlebury.edu/stereo/, 2009
[9] Sick LMS 200 http://www.sick.com, 2008
[10] Swissranger SR-3000 http://www.swissranger.ch, 2008
[11] PMDTech PMD[vision] 19k http://www.pmdtec.com, 2008
[12] A.R.T. ARTtrack system, see http://www.ar-tracking.com,2009
[13] NDI Polaris, see http://www.ndigital.com, 2009
[14] Faro Platinum arm, see http://www.faro.com, 2009
[15] RSI MicroScribe, see http://www.rsi-gmbh.de, 2009

Robotic manipulators are actuated measurement systems and are traditionally used for automatic scanning. However, robotic manipulators allow also for hand-guided operations. Recent concepts pursue approaches towards a robotic co-worker.
Algorithms for pose estimation from the range sensor data are registration methods, e.g. the real time system presented by Rusinkiewicz et.al. [RHHL02], or image-based ego motion estimators that uses the camera images to calculate the movement of the device, e.g. the monocular method of Burschka [Bur06], among others.
In the following sections, the range sensor geometry, the view alignment and transformation to 3D points using the sensor pose and the challenges in data synchronization between range sensor and pose sensor are further analyzed and a general description is derived.

2.2 Range Sensor Description

In this section a geometrical description of range sensors is evolved. A detailed classification of a range sensor concerning the type of image coordinate system and a description of the geometric mapping to Cartesian space w.r.t. its type is introduced. Further, geometrical per-point attributes are derived from this standardized description.

2.2.1 Geometric Mapping

The type of image coordinate system can be used to derive a standardized set of mappings for range images into Cartesian coordinates. In this work, four types covering the most range sensor types are identified:

- *Cartesian type* - The sensor emits parallel beams or uses a linear axis to actuate the device perpendicularly to the sensor beam.
- *Perspective type* - the sensor is described by a pin hole camera model. All beams intersect at one focal point.
- *Cylindrical type* - The sensor measures using a rotatory DoF and optionally a translatory DoF.
- *Spherical type* - The sensor measures using two rotatory DoF.

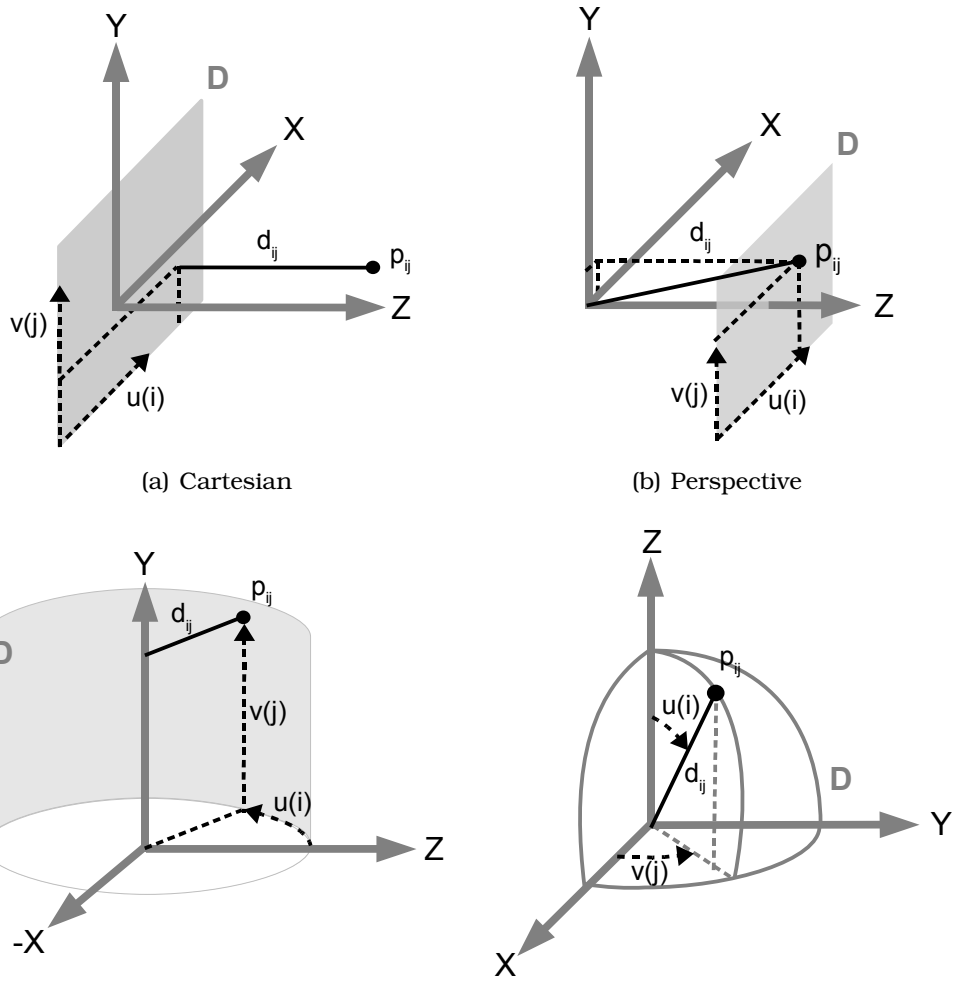
In Fig. 2.2 the relation between image coordinates and Cartesian space for each type is illustrated. In the following, a basic description of range images is introduced and the mapping of a distance pixels to Cartesian space for each of the four types is derived.

Let $\mathbf{D}$ denote the $(n \times m)$-matrix of a range image and let d_{ij} be a distance pixel, i.e. the value of $\mathbf{D}$ at the i-th row and j-th column within the working range of the sensor

$$0 < d_{\min} \leq d_{ij} \leq d_{\max} \ .$$

It is assumed that the coordinates of the range image are equidistant w.r.t. the image coordinate system. Hence, the grid locations or physical pixel coordinates $u(i)$ and $v(i)$ of a distance pixel d_{ij} in the image coordinate system can be described by the offsets u_0, v_0 and the sample widths $\Delta u, \Delta v$,

$$\begin{aligned} u(i) &= u_0 + i \cdot \Delta u; \quad i \in \mathbb{N}_m \\ v(j) &= v_0 + j \cdot \Delta v; \quad j \in \mathbb{N}_n \ , \end{aligned}$$



(a) Cartesian (b) Perspective

(c) Cylindrical (d) Spherical

Figure 2.2: Relation between image coordinates and Cartesian space: For each of the four image coordinate system types, the mapping to Cartesian space is illustrated, as defined in Equations (2.2)-(2.5). The image coordinates are defined by the u-, v-, and distance-directions w.r.t the xyz-directions of the Cartesian space. The image plane in Cartesian space is illustrated as gray region. Further, the Cartesian point p_{ij} is shown for an exemplary pixel d_{ij} with grid locations $u(i)$ and $v(j)$.

with the domains

$$\mathbb{N}_m = \{0, \ldots, m-1\}$$
$$\mathbb{N}_n = \{0, \ldots, n-1\} .$$

Generally, the mapping of a distance value d at the grid coordinates (u, v) into Cartesian space is

$$\mathbf{p} : \mathbb{R}^3 \to \mathbb{R}^3 \, ; (u, v, d) \mapsto \mathbf{p}(u, v, d) \ . \tag{2.1}$$

The mapping depends on the geometric type of the sensor. The mapping for each of the previously defined four types is given by:

$$\text{Cartesian:} \quad \mathbf{p}(u,v,d) := \begin{pmatrix} u \\ v \\ d \end{pmatrix} \tag{2.2}$$

$$\text{Perspective:} \quad \mathbf{p}(u,v,d) := \begin{pmatrix} d \cdot u \\ d \cdot v \\ d \end{pmatrix} \tag{2.3}$$

$$\text{Cylindrical:} \quad \mathbf{p}(u,v,d) := \begin{pmatrix} d \cdot \sin u \\ v \\ d \cdot \cos u \end{pmatrix} \tag{2.4}$$

$$\text{Spherical:} \quad \mathbf{p}(u,v,d) := \begin{pmatrix} d \cdot \sin u \cos v \\ d \cdot \sin u \sin v \\ d \cdot \cos u \end{pmatrix} \tag{2.5}$$

The corresponding Cartesian point $\mathbf{p}_{ij}$ of a distance pixel $d_{ij} \in \mathbf{D}$ at the i-th row and j-th column can be abbreviated by

$$\mathbf{p}_{ij} := \mathbf{p}(u(i), v(j), d_{ij}) \ .$$

This relation between $\mathbf{p}_{ij}$ and d_{ij} is further illustrated in Fig. 2.2.
Finally, the set of mapped points $\mathcal{P}_\mathbf{D}$ of the range image $\mathbf{D}$ is given by

$$\mathbf{D} \mapsto \mathcal{P}_\mathbf{D} := \{\mathbf{p}_{ij} \in \mathbb{R}^3 | i \in \mathbb{N}_m, j \in \mathbb{N}_n\} \ .$$

The two angles of the spherical coordinate system have (u,v)-ordering. A second spherical type with (v,u)-ordering is required for completeness of the classes, because the two types are not convertible to each other without losing the local ordering of the range image. This additional type is detailed and used in Chapter 7.

2.2.2 Line-of-Sight and Surface Normal

The surface normals for every distance pixel can be estimated directly from the range image. However, this image-based surface normal is not used in this work, since a calculation is limited to 2D range images and can result in bad estimates due to sparse or noisy measurements. This especially applies when large distances are measured, as the distance between two adjacent sample points is high and thus the sample density is low.
However, the line-of-sight (LoS) in Cartesian space for each point $\mathbf{p} \in \mathcal{P}_\mathbf{D}$ is required in the surface reconstruction for estimation of a surface normal from a multi-view point set. In detail, the line-of-sight $\mathbf{s}$ of a pixel with distance value d and grid coordinates (u,v) is the normalized connection vector between the corresponding Cartesian coordinate $\mathbf{p} = \mathbf{p}(u,v,d)$ and the point-specific ray origin $\mathbf{o} = \mathbf{p}(u,v,0)$,

$$\mathbf{s} := \frac{\mathbf{p} - \mathbf{o}}{||\mathbf{p} - \mathbf{o}||} \tag{2.6}$$

Substituting point $\mathbf{p}$ in Equation (2.6) by Equations (2.2)-(2.5) results in a rapidly computable representation of the line-of-sight $\mathbf{s} = \mathbf{s}(u,v)$ depending only on the grid

coordinates (u, v) for each geometric type:

$$\text{Cartesian:} \quad \mathbf{s}(u,v) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\text{Perspective:} \quad \mathbf{s}(u,v) = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cdot \frac{1}{\sqrt{u^2+v^2+1}}$$

$$\text{Cylindrical:} \quad \mathbf{s}(u,v) = \begin{pmatrix} \sin u \\ 0 \\ \cos u \end{pmatrix}$$

$$\text{Spherical:} \quad \mathbf{s}(u,v) = \begin{pmatrix} \sin u \cos v \\ \sin u \sin v \\ \cos u \end{pmatrix}$$

The above equations show the geometric differences of the different image coordinate systems. Translatory axes result in a line-of-sight that is independent of the pixel location. This characteristic is reflected in the Cartesian geometry and in the translatory axis of the cylindrical geometry. In contrary, rotational axes have a single center from which the distance rays originate, as seen e.g. in the spherical geometry. The perspective geometry is a special case, as here the distance value d of a pixel does not represent the length of the corresponding point from its origin. Thus, the line-of-sight must be normalized w.r.t. the grid location. Hence, the relation between a distance pixel d at the grid coordinates (u, v) and its corresponding sample point $\mathbf{p}(u, v, d)$, as principally defined in Equation (2.1), in Cartesian space can be alternatively expressed by the ray equation

$$\mathbf{p} = \mathbf{o} + d_{ray}\,\mathbf{s} \ , \tag{2.7}$$

with the ray length

$$d_{ray} = \begin{cases} d \cdot \sqrt{u^2+v^2+1} & \text{for perspective type} \\ d & \text{for all other types} \end{cases} .$$

Both the origin and the line-of-sight depend on the grid coordinates only, not on the distance.

2.2.3 Reference Sample Density

For surface reconstruction, the generated sample density or spatial resolution of the sample points on the surface is more interesting than the range image resolution, because the surface has to be sampled with sufficient density to enable a proper reconstruction. The sampling density depends on sensor resolution, local shape of the surface, and field of view of the sensor relative to the surface. The sample density potentially decreases with an increasing distance to the object and also with increasing grazing angles between the measurement rays and the surface normal. Hence, the **reference sample density** δ attribute is defined in this work as the sample density on a reference surface at a measured distance d. Thus, it is used as expected density at a sample point $\mathbf{p}$ produced by a single view or range image.
Let d_i denote the i-th distance measurement of a 1D range image with the corresponding sample point $\mathbf{p}_i$ and line-of-sight $\mathbf{s}_i$. Further, let Δ denote the sample width w.r.t.

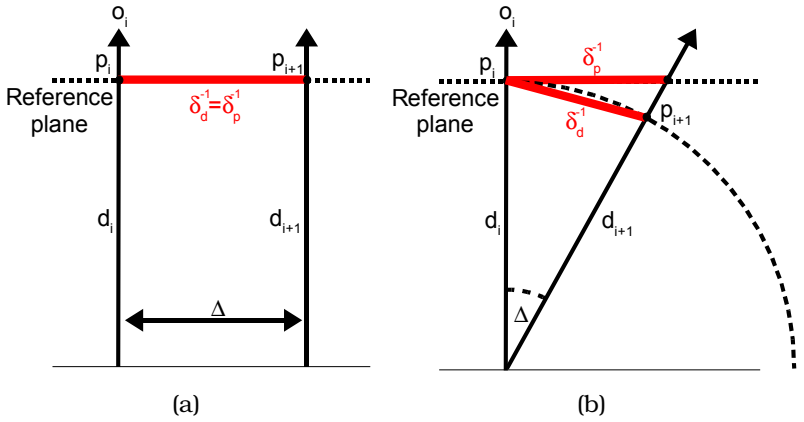


Figure 2.3: Relation between the two definitions of reference sample density: both definitions are equal for parallel measurement rays (a), both depend on the angular sample width Δ in the case of a central ray origin (b).

the type of image coordinate system. The reference sample density δ at point $\mathbf{p}_i$ is the reciprocal of the distance between $\mathbf{p}_i$ and the point $\mathbf{p}_{i+1}$ of the direct neighboring pixel on a reference surface,

$$\delta := \frac{1}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|} \,. \tag{2.8}$$

Consequently, the reciprocal δ^{-1} is the distance between two adjacent sample points on the reference surface. In the following, this distance is used to illustrate the definition of reference sampling density and in Chapter 3 it is used to parametrize the *RT-SSR* method.
Two assumptions for the definition of the reference surface and thus for the reference sample density are compared:

- Reference sample density δ_p on a tangent plane:
  Both sample points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are measurements on the same reference plane, which is perpendicular to the line-of-sight $\mathbf{s}_i$
- Reference sample density δ_d at equidistant measurements:
  Both sample points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ result from equidistant distance measurements $d_i = d_{i+1} = d$

In Fig. 2.3 both approaches are illustrated for a sensor with parallel measurements rays and for one with measurements from a central origin. Both definitions result in the same resolution for parallel measurement rays,

$$\delta_p = \delta_d = \Delta^{-1} \,, \tag{2.9}$$

with a translatory sample width Δ. In the case of a central ray origin, the distances are related by

$$\frac{\delta_p}{\delta_d} = \frac{\cos\Delta}{\cos\frac{\Delta}{2}} \,, \tag{2.10}$$

with an angular sample width $\Delta \in (0°, 90°)$. The derivation of the above equation is presented in Appendix B.
The definition of sample density on a reference plane can be extended to the density $\delta_{p\alpha}$ on a plane that is tilted by an angle α to the original plane, representing the change of sample density at an increasing grazing angle. The relation between $\delta_{p\alpha}$ and δ_p is

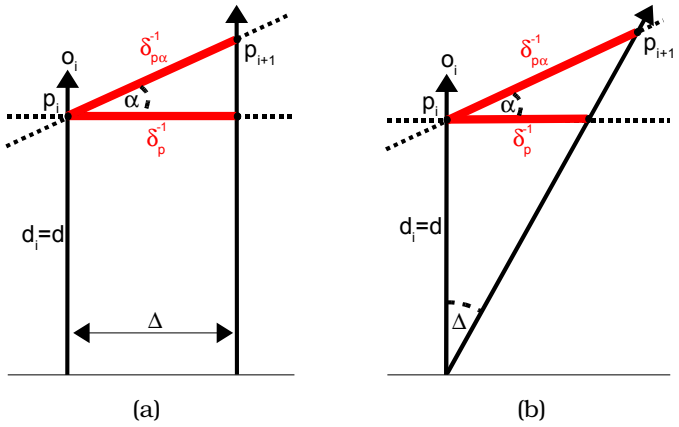


Figure 2.4: Sample density on a tilted reference plane: The inverse densities or distances respectively δ_p^{-1} and $\delta_{p\alpha}^{-1}$ are illustrated for parallel rays (a) and for a central origin (b). In both cases the density decreases with higher values of α.

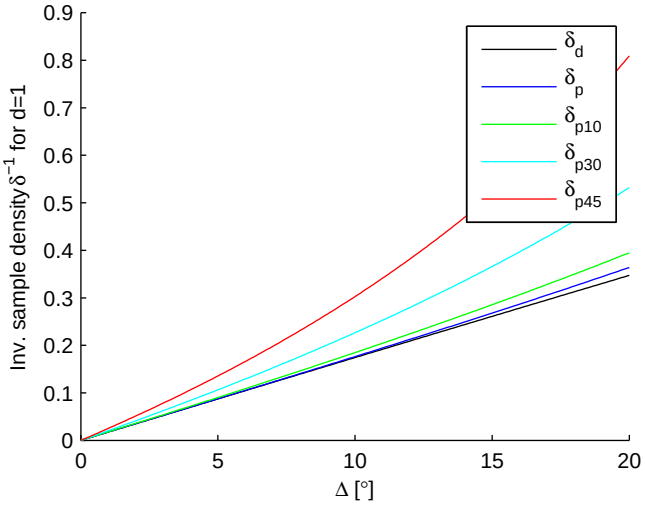


Figure 2.5: Comparison of the different sample densities for a rotatory DoF: The plot shows δ_d, δ_p, and $\delta_{p\alpha}$ for $\alpha = \{10°, 30°, 45°\}$ with respect to the angular sample width $\Delta = [0°, 20°]$ and at a distance $d_i = 1$.

illustrated in Fig. 2.4. The relation between the definitions in the case of parallel measurement rays $\delta_{p\alpha}$ is

$$\delta_{p\alpha} = \delta_p \cos\alpha = \delta_d \cos\alpha \ ,$$

and the equation for a central ray origin is

$$\delta_{p\alpha} = \delta_p \frac{\cos(\alpha+\Delta)}{\cos\Delta} = \delta_d \frac{\cos(\alpha+\Delta)}{\cos\frac{\Delta}{2}} \ .$$

In Fig. 2.5, the reference densities for δ_d, δ_p, $\delta_{p10°}$, $\delta_{p30°}$, and $\delta_{p45°}$, are plotted for an angular sample width $\Delta = [0°, 20°]$ and at a distance $d_i = 1$. The sample density is constant for parallel rays and decreases with an increasing distance for a central origin. The different definitions are related by a constant factor and differ only minimally for small values of Δ. Hence, the sample density whose calculation can be most simplified w.r.t. the description of a range image is used for generating attributes for every point $\mathbf{p} \in \mathcal{P}_\mathbf{D}$. In this work, the sample density at equidistant measurements is

used as per-point attribute

$$\delta := \delta_d$$

because of its fast computability.
The reference sample density δ_u, δ_v for the $u-$ and $v-$directions of a sample point $\mathbf{p}_{ij}$ that has been generated from the i-th row and j-th column of a range image $\mathbf{D}$ can be calculated by modifying Equation (2.8) to

$$\delta_u := \frac{1}{\|\mathbf{p}_{i+1,j} - \mathbf{p}_{i,j}\|} \tag{2.11}$$

$$\delta_v := \frac{1}{\|\mathbf{p}_{i,j+1} - \mathbf{p}_{i,j}\|} \ . \tag{2.12}$$

The overall expected density δ is the minimum of the directional densities

$$\delta := \min\{\delta_u, \delta_v\} \ .$$

Further, the directional densities δ_u and δ_v can be expressed as functions that directly depend on the measured distance d and the geometric sensor parameters Δu and Δv, w.r.t. the geometric type of the sensor. Therefore, the sample points in Equation (2.11) and Equation (2.12) are substituted by the type specific mappings from Equations (2.2)-(2.5). This results in the following calculations for the different geometric types:

$$\text{Cartesian:} \quad \delta = \min\{|\Delta u^{-1}|, |\Delta v^{-1}|\} \tag{2.13}$$

$$\text{Perspective:} \quad \delta = \min\{(d|\Delta u|)^{-1}, (d|\Delta v|)^{-1}\} \tag{2.14}$$

$$\text{Cylindrical:} \quad \delta = \min\{(2d|\sin\frac{\Delta u}{2}|)^{-1}, |\Delta v^{-1}|\} \tag{2.15}$$

$$\text{Spherical:} \quad \delta = \min\{(2d|\sin\frac{\Delta u}{2}|)^{-1}, 2d|\sin u \ \sin\frac{\Delta v}{2}|)^{-1}\} \tag{2.16}$$

The full derivation of the above equations is given in Appendix B.
For a spherical geometry the density not only depends on the sample widths Δu and Δv but on the absolute pixel position u. This reflects the fact that the two DoF of the spherical geometry are coupled and the density increases ($\delta_u \mapsto \infty$) at the domes of the spherical workspace. The perspective geometry is a special case, as the ray length is not equal to the distance (see the previous section). This type has a central ray origin but the sample widths are the distances between two pixels at $d = 1$ and not an angular value.

2.3 View Alignment and Synchronization

For the alignment of measured range images or views in real time, the pose of the scanner device at the range image's measurement time is required. In this work the measurement of the pose is modeled by a principal pose sensor unit, depicted in Fig. 2.1 on page 10. However, pose measurement is not necessarily performed synchronously to the acquisition of range images. The problem of assigning the correct pose for every range image is discussed in this section and a general synchronization and interpolation concept is introduced.

2.3.1 Geometric Description of Pose Data

The pose of a rigid body in the Cartesian space is mathematically described by a rotation and a translation, denoted as **rigid motion transformation**. Unlike the variety in system specific representations of such transformations, the representation as a 4×4 homogeneous matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}' & 1 \end{pmatrix}.$$

with the 3×3 rotational matrix $\mathbf{R}$ and the translational vector $\mathbf{t}$ is used. This notation is commonly used in the field of robotics and computer graphics. Detailed information concerning transformations and homogeneous coordinates are e.g. given by Craig [Cra89] or Akenine-Moeller [AMH02].
Let ${}^{w}\mathbf{T}_{s}(\tau)$ denote the pose of the range sensor in a global coordinate system, i.e. the transformation from the Cartesian sensor coordinate system (s) to the world coordinate system (w) at a measurement time τ. This pose is typically not measurable directly, e.g. the native coordinate system of a camera is not accessible, as it is inside the system. Instead, a local coordinate system (l) is measured by the transformation ${}^{l}\mathbf{T}_{w}(\tau)$ that has a rigid connection to the sensor coordinate system. An additional transformation ${}^{l}\mathbf{T}_{s}$ connects the sensor coordinates to the local coordinate system. The transformation is usually estimated during system calibration and is specific to the systems's particular combination of a range sensor and a pose sensor.
Consequently, a sample point $\mathbf{p}_{s}(\tau) \in \mathcal{P}_{\mathbf{D}}$ at a measurement time τ is transformed to a point $\mathbf{p}$ in global space by

$$\begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = {}^{w}\mathbf{T}_{s}(\tau) \begin{pmatrix} \mathbf{p}_{s}(\tau) \\ 1 \end{pmatrix} = {}^{w}\mathbf{T}_{l}(\tau)\, {}^{l}\mathbf{T}_{s} \begin{pmatrix} \mathbf{p}_{s}(\tau) \\ 1 \end{pmatrix} . \tag{2.17}$$

Accordingly, the corresponding line-of-sight $\mathbf{s}_{s}$ is transformed to $\mathbf{s}$ by

$$\begin{pmatrix} \mathbf{s} \\ 0 \end{pmatrix} = {}^{w}\mathbf{T}_{s}(\tau) \begin{pmatrix} \mathbf{s}_{s} \\ 0 \end{pmatrix} .$$

2.3.2 Sensor Synchronization and Data Labeling

The measurements of pose and range image can operate at different frequencies or are phase shifted. Phase shift means that measurements are performed at the same frequency but the measurement time differs by a constant shift. The problem of different cycle rates can occur when an external device is used for pose measurement, e.g. a robot or an optical tracking system. External devices usually have an internal measurement frequency independent of the range sensor's cycle rate. Even if the devices theoretically have the same cycle rate, in practice, the measurement frequencies of independent systems always differ slightly, resulting in a continuous drift of the measurements.
The cycle time and the measurement time of both sensors for each period can be coupled by externally triggering the hardware components, denoted as **sensor synchronization**. This trigger is either implemented as electrical pulse (**hardware synchronization**) or as a communication package (**software synchronization**). Not all hardware components support an external trigger and therefore this concept is not applicable to all scanner systems.
Beside the different measurement times, the communication and subsequent processing of raw measurement data for calculation of pose and range image result in a

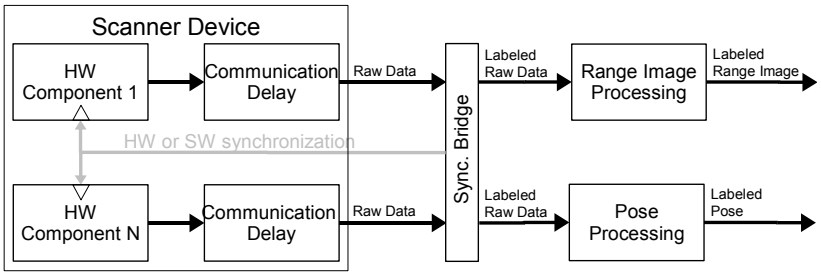


Figure 2.6: Concept of sensor synchronization and data labeling: The raw data from the hardware components of the scanner system is transfered to the host system with possible communication delay. The synchronization bridge labels the raw data and synchronizes the measurement times in the hardware components, if applicable. Finally, the labeled data streams are used to calculate or to provide range images and poses.

delayed data availability. Hence, for the alignment of a range image using the measured poses, a consistent **temporal data labeling** of all measured data is mandatory, so that the range images and pose data can be matched.
In Fig. 2.6, the general concept of data labeling and possible sensor synchronization is illustrated. All incoming data streams of the hardware components are labeled by a **synchronization bridge**, providing global time stamps for all data. If the respective component supports this feature, the hardware components are also synchronized by the bridge. Accordingly, range images and poses are calculated from the respective raw data streams. Here, calculation consists of at least a marshalling step that prepares the raw data for the transfer to subsequent processing. This general concept has already been described by Bodenmüller et al. [BSSH07].
For correct labeling of the data the communication and processing delay must be known. If all data is measured at the same rate i.e. the system is synchronized, only the phase shift and the delay in cycle rates is required. Consequently, if synchronization is not available, the system is less robust against a jitter in the delay, resulting in less accurately aligned 3D point data. If labeling is performed on a distributed system, it is important that the time stamp used for labeling is generated by a system-global clock. Bodenmüller et al. [BSSH07] uses a global consecutive counter with an additional offset as time stamp, generated by a hardware component and distributed to all systems. Another approach is the direct use of a real time clock i.e. the use of an absolute time as time stamp. For distributed systems, the clocks have to be synchronized using the *Network Time Protocol (NTP)* or *Precision Time Protocol (PTP)* service.

2.3.3 Pose Interpolation and Stream Generation

Synchronization and data labeling allow for matching pose data and range images temporally. In order to transform the range image into a set of 3D points, the sensor's pose at the measurement time of each pixel has to be determined. Depending on the measurement principle of the range sensor, the distances in a range image are not necessarily measured at the same time but sequentially. Consequently, the following measurement modes must be distinguished:

- **synchronous** - All data is acquired at once, i.e. at the same time.
- **line-sequential** - The range image is measured stripe-wise in u- or v-direction.
- **field-sequential** - The range image is measured pixel-by-pixel, either row-major or column-major.

The pose has to be determined once per range image, for every stripe, or for every pixel respectively, depending on the measurement mode. In the following, an interpolation scheme is presented that determines the required pose.
Let $d_{ij}(\tau_m)$ denote a distance pixel measured at the time τ_m and let ${}^w\mathbf{T}_l(\tau_m)$ be the corresponding pose that has to be determined. Different approaches to estimating the desired pose ${}^w\mathbf{T}_l(\tau_m)$ are possible. The choice of method depends on the temporal relation between pose and range images. If both are measured at the same rate and without any phase shift, the pose at time τ_m can directly be extracted from the stream of labeled pose data. If the rate differs or a phase shift is present, the pose at time τ_m has to be interpolated from the measured poses. The suitability of an interpolation method depends on the frequencies of range and pose measurement and on the movement rate of the sensor between two cycle periods. In the following three common interpolation strategies are discussed:

Nearest Pose The nearest pose method uses the pose that is measured temporally closest to the range image. This method only results in reliable corresponding poses if the time between measurements of range image and pose is small in relation to the movement rate of the device. This applies if either the frequency of pose measurement is significantly higher than the frequency of the range sensor or if both measurements are synchronous or with only a small phase shift.

Linear Pose Interpolation The linear pose interpolation is more accurate than the nearest pose strategy, yet it is still fast computable. It allows for an accurate pose interpolation at a faster movement of the scanner device w.r.t. the measurement rate and for phase shifted pose signals.
Let τ_m denote the measurement time of range image $\mathbf{D}(\tau_m)$ and let τ_k and τ_{k+1} denote the measurement time of k-th and (k+1)-th pose measurements ${}^w\mathbf{T}_l(\tau_k)$ and ${}^w\mathbf{T}_l(\tau_{k+1})$ with

$$\tau_k \leq \tau_m \leq \tau_{k+1}.$$

Further, let $\Delta\tau_m$ denote the normalized time of range measurement defined by

$$\Delta\tau_m = \frac{\tau_m - \tau_k}{\tau_{k+1} - \tau_k} \in [0,1] \quad .$$

The connecting transformation between the poses ${}^w\mathbf{T}_l(\tau_k)$ and ${}^w\mathbf{T}_l(\tau_{k+1})$ is denoted as

$$\Delta\mathbf{T} = \begin{pmatrix} \Delta\mathbf{R} & \Delta\mathbf{t} \\ \mathbf{0}' & 1 \end{pmatrix} := {}^w\mathbf{T}_l^{-1}(\tau_k) \ {}^w\mathbf{T}_l(\tau_{k+1}) \ .$$

The rotation $\Delta\mathbf{R}$ can be transformed into an axis-angle representation (see Appendix A.2):

$$\Delta\mathbf{R} \mapsto (\mathbf{n}_{\Delta\mathbf{R}}, \alpha_{\Delta\mathbf{R}}) \quad \alpha \in [0^\circ, 180^\circ) \ .$$

The linear interpolated pose $\mathbf{T}_{\tau_m}$ at the range measurement time τ_m is

$${}^w\mathbf{T}_l(\tau_m) = {}^w\mathbf{T}_l(\tau_k) \ \begin{pmatrix} \Delta\mathbf{R}_{\tau_m} & \Delta\mathbf{t}_{\tau_m} \\ \mathbf{0}' & 1 \end{pmatrix}$$

with the relative translation

$$\Delta\mathbf{t}_{\tau_m} = \Delta\mathbf{t}\Delta\tau_m$$

and the relative rotation

$$\Delta\mathbf{R}_{\tau_m} = \mathbf{R}_{aa}(\mathbf{n}_{\Delta\mathbf{R}}, \alpha_{\Delta\mathbf{R}} \cdot \Delta\tau_m)$$

with the transformation of an axis-angle pair to a rotation matrix $\mathbf{R}_{aa}(\mathbf{n}, \alpha)$ (see Appendix A.2).

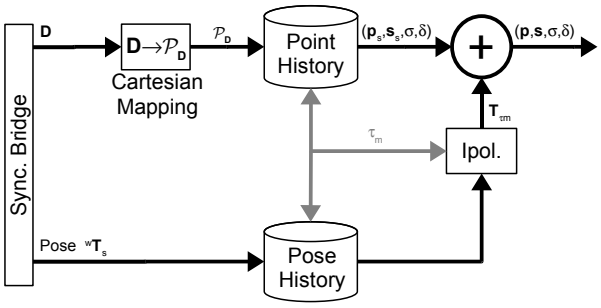


Figure 2.7: Concept of view alignment and serialization towards a 3D point stream: The received labeled images are transformed to 3D points in the local sensor coordinate system and stored in a point history. The labeled poses are also stored in a history module. If for a point all necessary poses are available the required pose is interpolated and the point is transformed to global space, using the interpolated pose.

Advanced Pose Interpolation The linear pose interpolation scheme can result in poor estimates if the movements between the poses are significant i.e. if the measurement rate of the pose sensor is slow w.r.t. the movement rate of the scanner device. In this case, advanced pose interpolations by higher-order polynomials or splines have to be used. However, these concepts are not used in this work, as the cycle rate of the pose sensor is assumed to always be sufficiently high.

Fig. 2.7 illustrates the processing concept of pose interpolation and view alignment. The incoming labeled poses are stored in a history queue and are used as sources for the interpolation unit. The labeled range images are transformed into 3D points as described in Equations (2.2)-(2.5). Additionally, corresponding per-point features are calculated as presented in Section 2.2. An additional attribute $\tilde{\sigma}$ represents the estimated accuracy of the point and is discussed in Section 2.4. The 3D points and the corresponding attributes are also stored in a point history queue. The point history fulfills two functions: First, the set of 3D points per range image is transformed into a serial stream of single 3D points. Second, the subsequent global alignment of a 3D point can be delayed until all necessary poses are available. The interpolation unit is fed with the current pose history, providing the interpolated pose at the required time τ_m. Finally, a globally aligned 3D point and corresponding attributes are calculated from the local data from the point history and the corresponding pose from the interpolation unit, as described in Equation (2.17).
Generally, an extrapolation strategy for pose estimation is an alternative to interpolation. Extrapolation predicts the required pose from the N last poses. Thus, delays in processing due to a delayed pose stream can be avoided. This strategy is preferred in the context of feedback control, because additional delays can cause system instabilities. However, additional delay is uncritical in the context of a visual feedback for manual scanning. Hence, the more simple and accurate interpolation strategy is used in this work.

2.4 Measurement Errors and Accuracy

The accuracy of every scanner system is limited due to measurement errors. In single-view applications and in methods that process range images individually only errors in the distance measurements are relevant. When multiple range images are aligned and merged into a single point set, the accuracy of the pose estimation must also be considered. In off-line applications the misalignment caused by a pose error can be

minimized by a preceding optimization step, e.g. bundle adjustment or point registration. However, manual scanning with a real time data processing, a minimization of pose errors is not possible, thus they influence the measurements and the accuracy of the system.

2.4.1 Range Sensor Errors

While the geometry of a range sensor can easily be generalized, measurement errors highly depend on the physical principle of the range measurement, on the properties of the measured object (e.g. reflectivity, micro structure), and on environmental settings. However, object properties and the environment are usually unknown.
The descriptions of the errors of range sensors are typically specialized to a certain sensor type or to the application the system is used for. In the work of Curless [Cur97], the physical error sources for active triangulation systems are summarized in the context of 3D modeling, while Fuchs and Hirzinger [FH08] list error sources of on-chip ToF systems in the context of optimal system calibration. More general sensor models for autonomous robot work cell exploration are examined in the work of Suppa [Sup07]. Here, the models are general w.r.t. the measurement principle but specialized w.r.t. the application, i.e. the model is optimized for deciding whether a volume element is free or occupied considering a noisy sensor.
A range image is the synchronous or sequential measurement of its distance pixel, as already stated in the previous sections. Hence, potential errors and limitations can generally be modeled per pixel. The overall measurement error of each distance pixel can be divided into a positional error and a distance error w.r.t. the geometric description in Section 2.2.1. Here, the distance error is the composition of all physical errors and technical limitations of the measurement principle applied, the sensor hardware and the surface properties. This includes e.g. sensor noise, quantization errors, and effects caused by the reflectivity and micro structure of the scanned surface. A positional error reflects an error between the measured or expected[16] grid coordinates of a distance pixel and the real values. Examples for this kind of error are the inaccurate measurement of the position of a rotating axis, or a distortion in the camera image.
In this work, it is assumed that a range sensor is always intrinsic calibrated, i.e. systematic errors are compensated. Consequently, the relation between the real pixel vector $(u, v, d)^T$ and the measured values $(\tilde{u}, \tilde{v}, \tilde{d})^T$ can be described as

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{d} \end{pmatrix} = \begin{pmatrix} u \\ v \\ d \end{pmatrix} + \begin{pmatrix} e_u \\ e_v \\ e_d \end{pmatrix} \quad .$$

In most range sensors, the error e_d increases with an increasing distance d, whereas the positional errors (e_u, e_v) are typically independent of the distance. The overall error is often modeled as an ellipsoidal Gaussian distribution that increases with the distance, e.g. in the work of Hirschmüller [Hir03] for a stereo vision system.

2.4.2 Pose Sensor Errors

A pose error, i.e. a position error or orientation error, influences the entire range image; contrary to range sensor errors, which are modeled pixel-wise. Pose errors

[16] In many systems, the grid coordinates are not measured but represent a physical coordinate, e.g. a pixel in a camera.

can have very different characteristics depending on the sensor. External measurement devices, e.g. coordinate measurement machines or optical tracking systems, are independent of the range measurement, but they typically have complex error distributions that potentially depend on the absolute pose and the movement speed of the scanner device. Contrary, pose estimation methods that use the same sensor hardware as the range sensor have pose errors that are correlated with the depth measurement.
In the work of Scott et al. [SRR02], the impact of position and orientation errors on the precision and sensor coverage is described, in the context of next best view planning. Following this argumentation, pose errors can be divided into positioning errors, axis errors, and twist errors. A pose error always results in the misalignment of the corresponding range image, decreasing the overall accuracy of the merged point set.
Generally, the pose error can be modeled as a location error and an orientation error w.r.t. the measured coordinate frame. Hence, the measured pose ${}^{w}\tilde{\mathbf{T}}_{l}$ can be described w.r.t. to the real pose ${}^{w}\mathbf{T}_{l}$ by

$$ {}^{w}\tilde{\mathbf{T}}_{l} := {}^{w}\mathbf{T}_{l} \begin{pmatrix} R_{rpy}(\mathbf{e}_r) & \mathbf{e}_t \\ \mathbf{0}' & 1 \end{pmatrix} , $$

with a location error $\mathbf{e}_t$ and a rotation error $\mathbf{e}_r$ in each local axis. The latter is modeled as *roll-pitch-yaw (rpy)* angles (see Appendix A.2). Moreover, the extrinsic calibration ${}^{l}\mathbf{T}_{s}$ is typically determined experimentally, e.g. by an eye-in-hand calibration[17]. Hence, the estimated calibration ${}^{l}\tilde{\mathbf{T}}_{s}$ possibly differ from the real transformation ${}^{l}\mathbf{T}_{s}$. Consequently, the (measured) transformation from sensor coordinates to global space ${}^{w}\tilde{\mathbf{T}}_{l}$ is given by

$$ {}^{w}\tilde{\mathbf{T}}_{s} := {}^{w}\mathbf{T}_{l} \begin{pmatrix} R_{rpy}(\mathbf{e}_r) & \mathbf{e}_t \\ \mathbf{0}' & 1 \end{pmatrix} {}^{l}\tilde{\mathbf{T}}_{s} . $$

This leads to the conclusion that a pose error implicates that the measured position and view direction of the range sensor differ from the real values. Hence, an area on the object's surface is measured that is not the expected part. In the next section, this coupling between pose errors and distance measurement is further analyzed.

2.4.3 Overall Error and Per-Point Accuracy

In the previous sections, the errors of a scanner system are described component-based. However, in the context of streaming surface reconstruction in 3D, a scalar per-point quality criterion that represents the overall accuracy of a single distance measurement is desired. Hence, the measurement errors of range and pose sensor must be modeled as a combined error for each sample point. Thus, the description of a sample point as a measurement ray in global space is used to model an overall measurement error, the error in the ray length is used as a metric for a per-point quality.
As introduced in Equation (2.7), a sample point $\mathbf{p}_s$ in the Cartesian sensor space can be described by a measurement ray of length d_{ray} that starts in the ray origin $\mathbf{o}_s$ and has the direction $\mathbf{s}_s$. The corresponding point $\mathbf{p}$ in global space is calculated by applying the transformation ${}^{w}\mathbf{T}_{s}$, described in Equation (2.17). Hence, the globally aligned ray equation is given by

$$ \mathbf{p} = \mathbf{o} + d_{ray}\,\mathbf{s} , $$

[17] See e.g. the work of Strobl and Hirzinger [SH06] for details on eye-in-hand calibration methods.

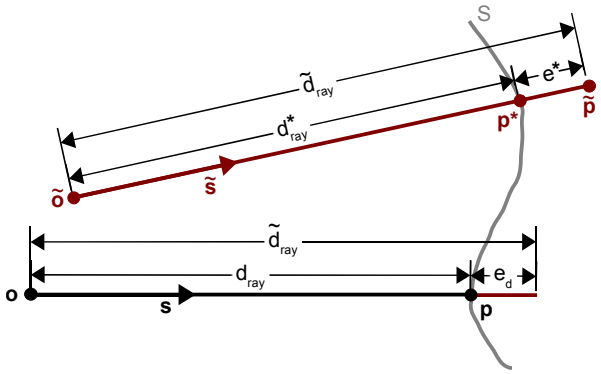


Figure 2.8: Difference between real and measured sample point: A measured ray origin $\tilde{\mathbf{o}}$ and ray direction $\tilde{\mathbf{s}}$ that differ from the real values $\mathbf{o}$ and $\mathbf{s}$ cause a erroneous ray length $\tilde{d}_{ray}$ with an error e^* that generally depends on all error-prone variables and the local shape of the surface. Consequently, the error e^* of the measured ray length differs typically from the distance measurement error e_d.

with the global origin

$$\begin{pmatrix} \mathbf{o} \\ 1 \end{pmatrix} = {}^w\mathbf{T}_s \begin{pmatrix} \mathbf{o}_s \\ 1 \end{pmatrix}$$

and ray direction

$$\begin{pmatrix} \mathbf{s} \\ 0 \end{pmatrix} = {}^w\mathbf{T}_s \begin{pmatrix} \mathbf{s}_s \\ 0 \end{pmatrix} \ .$$

As described in the previous sections, the measured distance, grid coordinates, and global pose are possibly affected by noise and thus are error-prone. These errors result in a measured ray origin $\tilde{\mathbf{o}}$, ray direction $\tilde{\mathbf{s}}$, and ray length $\tilde{d}_{ray}$ that differ from the real values. This results in a erroneous sample point

$$\tilde{\mathbf{p}} = \tilde{\mathbf{o}} + \tilde{d}_{ray}\,\tilde{\mathbf{s}} \ .$$

The ray's origin and direction are influenced by errors in the grid coordinates and the global pose, but are independent of errors in distance measurement. In contrast, the ray length is influenced by all errors, as it depends on the distance measurement and is also coupled to possible errors in ray origin and direction. This problem is illustrated in Fig. 2.8. An error-prone ray origin or direction results in a point $\mathbf{p}$ on the surface along the real ray ($\mathbf{o}$,$\mathbf{s}$) being sampled that does not correspond to the expected sample point $\mathbf{p}^*$ along the measured ray ($\tilde{\mathbf{o}}$,$\tilde{\mathbf{s}}$). As the distance is measured w.r.t. the real ray origin and direction, the measured distance does not correspond to the measured ray origin and direction. Let d^*_{ray} denote the distance of the ray from the measured origin $\tilde{\mathbf{o}}$ to the expected sample point $\mathbf{p}^*$ along the measured direction $\tilde{\mathbf{s}}$. The measured ray length $\tilde{d}_{ray}$ can be described as

$$\tilde{d}_{ray} := d^*_{ray} + e^* \ .$$

The ray error e^* integrates all errors caused by range sensor and pose measurement, thus it differs from the distance error e_d. Generally, the ray error e^* is a function of all error-prone variables and the unknown surface[18], i.e.

$$e^* = e(d, u, v, {}^w\mathbf{T}_s, S) \ .$$

[18]In detail, the error depends on the shape of the surface and its optical properties (e.g. reflectivity, micro-structure, etc.)

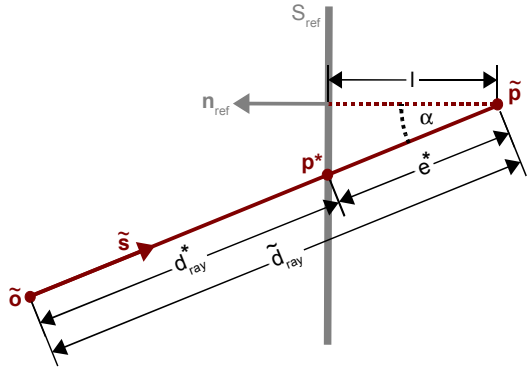


Figure 2.9: Measurement of the ray error e^* on a known reference plane: The shortest distance between the measured sample point $\tilde{\mathbf{p}}$ and the reference plane S_{ref} is calculated. The error e^* is thus given by the triangle spanned by the reference plane, the ray direction $\tilde{\mathbf{s}}$, and the plane normal $\mathbf{n}_{ref}$.

In this work, the ray error e^* is used to define a scalar quality attribute that can be calculated for every sample point. It is assumed that the scanner system is calibrated, i.e. systematic errors are compensated. The remaining error depends at least on the distance d and is thus modeled by a Gaussian zero-mean distribution

$$e^*(d) \sim N(0, \sigma(d)) \ ,$$

with the distance-dependent deviation $\sigma(d)$.

The distance-dependent deviation $\sigma(d)$ can be estimated experimentally by performing measurements from different viewpoints on a fixed reference plane with a known global pose. Let S_{ref} denote the reference plane, defined by the plane normal $\mathbf{n}_{ref}$ and the distance d_{ref},

$$S_{ref} := \{\mathbf{x} \in \mathbb{R}^3 \mid \langle \mathbf{n}_{ref}, \mathbf{x} \rangle - d_{ref} = 0 \ , \ d_{ref} > 0 \ , \ \|\mathbf{n}_{ref}\| = 1\} \ .$$

Further, let l denote the shortest distance between a measured sample point $\tilde{\mathbf{p}}$ and S_{ref},

$$l = \langle \mathbf{n}_{ref}, \tilde{\mathbf{p}} \rangle - d_{ref} \ ,$$

and let α denote the angle between the measured ray direction $\tilde{\mathbf{s}}$ and the plane normal $\mathbf{n}_{ref}$

$$\alpha = |\langle \mathbf{n}_{ref}, \tilde{\mathbf{s}} \rangle| \ .$$

Hence, the ray error $e^*(\tilde{\mathbf{p}})$ of the measured sample point $\tilde{\mathbf{p}}$ is given by

$$e^*(\tilde{\mathbf{p}}) = \frac{l}{\alpha} = \frac{\langle \mathbf{n}_{ref}, \tilde{\mathbf{p}} \rangle - d_{ref}}{|\langle \mathbf{n}_{ref}, \tilde{\mathbf{s}} \rangle|} \ .$$

This concept is illustrated in Fig. 2.9. The deviation $\sigma(d)$ can be estimated from the measured data by statistical methods, i.e. the measured data is sorted by the distance value and the deviation is calculated either by a sliding window technique or by partitioning the data. Finally, an analytic description for $\sigma(d)$ can be approximated by fitting a polynomial of order n through the calculated deviation values, resulting in a description of the type

$$\sigma(d) := \sum_{i=0}^{n} a_i p^i \ .$$

Exemplary results of this procedure using a certain scanner system, the *DLR Multi-sensory 3D Modeler*, is presented in Chapter 6.
Since the real distance is not known when measuring an arbitrary surface, it is assumed that the deviation of the measured distance $\sigma(\tilde{d})$ is a suitable approximation of $\sigma(d)$,

$$\sigma(\tilde{d}) \approx \sigma(d) \ .$$

The deviation $\sigma := \sigma(\tilde{d})$ is used in the following chapters as per-point accuracy attribute.

2.5 Summary and Discussion

This chapter analyzes the properties of manually operated 3D scanner systems. A manual scanner system is defined to consist of a range sensor, capturing an ordered set of distance pixels, and a pose sensor, measuring the pose of the scanner device in a global coordinate system. A general sensor model is derived, considering three aspects: geometric properties, synchronization, and accuracy.
The geometric properties of a range sensor are categorized concerning the type of mapping between image coordinates and Cartesian space. The calculation of global 3D points from a range image is derived for each of the four defined types, also considering the device pose and a possible extrinsic calibration. A set of per-point attributes is defined that is used in the subsequent reconstruction process.
The geometrical description that is introduced covers most types of range sensors, including 2D range sensors (e.g. structured light or stereo reconstruction systems) 1D sensors (e.g. light stripe sensor), and scalar sensor (e.g. single point measurement units). However, it is limited to sensors providing the data on a rectangular grid, with constant step width between neighboring pixels in the local sensor coordinate system. Hence, data with inhomogeneous ordering, e.g. using undistorted image coordinates, is not covered. A homogenization of the range image can be used to convert the image into the required format, however, this possibly decreases the accuracy of the measurement. It is also possible to use a different geometric model that allows for extracting the same attributes, as the subsequent surface reconstruction does not depend on the model itself, but requires only an input stream of 3D points with the introduced attributes attached, as shown in Fig. 1.2.
The temporal description covers the synchronization of measurements and the assignment of the correct pose to every range image or distance pixel. Here, a synchronization of all hardware components in combination with a labeling of all data is suggested. The labeling guarantees a correct assignment of all data, even in delayed or non-real-time processing, and is thus mandatory. The synchronization is optional and increases the robustness of the system. The labeling enables an interpolation of the correct pose for the required measurement time of the respective range image. The range images are transformed into a set of globally aligned 3D points, using the interpolated poses. The resulting 3D points and the additional per-point features are finally transfered to subsequent processing i.e. to the *RT-SSR* processing, as a serial stream.
The principal synchronization and labeling methods have to be adapted to the hardware and methodologies used. If the processing or communication delay have a large jitter, it is difficult to calculate the measurement time in relation to the time of reception. In this case, a synchronization of the hardware components is recommended. Hardware synchronization is more precise than software synchronization, because

the signal on a data bus can jitter, especially if the full bandwidth of the bus is used. Contrary, a software synchronization does not require additional cabling to all hardware components.
The description of accuracy for a manual scanner systems is covered by defining a scalar per-point quality attribute. The quality represents the distance-dependent deviation into the expected measurement direction. This metric implicitly integrates errors from range image measurement and pose measurement. The key advantage of this description is that it can be determined experimentally, e.g. by performing measurements on a known reference plane.
In summary, this chapter presents a general description of manual scanner systems and their characteristics. For each distance pixel of a measured range image, the coordinate of the resulting sample point as well as corresponding attributes describing the expected resolution, accuracy, and line-of-sight are calculated. The resulting stream of 3D points with corresponding attributes is used as input for the *RT-SSR* processing in the next chapter.

3
Streaming Surface Reconstruction

In this chapter, the **Real Time Streaming Surface Reconstruction (RT-SSR)** algorithm is described in detail. The goal is to build a dense triangle mesh by incrementally inserting 3D points from a real time stream. In most surface reconstruction algorithms, including the one presented here, it is essential to know the surface normal for every sample point. Consequently, surface normals have to be estimated before the mesh generation. The resulting reconstruction pipeline thus consists of two principal processing stages: The **normal estimation stage** and the **mesh generation stage**, as illustrated in Fig. 3.1:

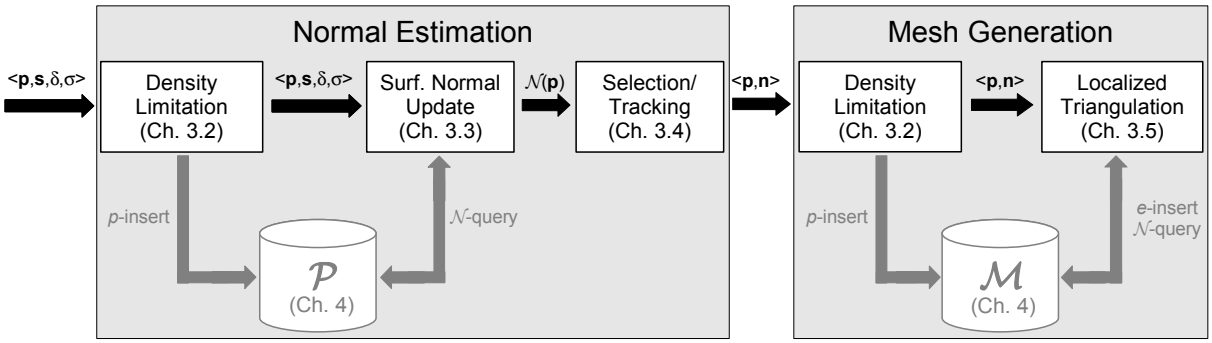


Figure 3.1: The *RT-SSR* process consists of two principle stages: Sample points are first processed in a normal estimation stage. Subsequently, points with an assigned surface normal are used for an incremental generation of a triangle mesh.

The incoming 3D points from the stream are defined by a coordinate p, a line-of-sight s, a reference sample density δ, and an expected deviation σ as described in Chapter 2. This point data is received in the **Density Limitation** step. The point density is limited here, so the sample density and the computational effort in successive stages can be controlled. The points that pass the limitation step are stored in a suitable data structure. The surface normals of this new point and the points it influences are estimated in the successive **Surface Normal Update** step. The estimated normals are validated in the **Selection and Tracking** module and are rejected if not plausible. Additionally, changes in the direction of the estimated normals are tracked and notified to successive stages. The outgoing stream of points with corresponding surface

normals enters the mesh generation stage. Again, the incoming points are stored as the vertices of the generated mesh. An additional **Density Limitation** controls the resolution of the final mesh. The stored vertices represent the vertices of the mesh that is generated and refined in the **Localized Triangulation**.

3.1 Geometric Definitions

This section introduces the geometric definitions used in the subsequent sections.

3.1.1 Point Sets

Definition

Let $\mathcal{P}$ denote the set of **sample points**

$$\mathcal{P} := \{\mathbf{p}_1, ..., \mathbf{p}_n \in \mathbb{R}^3\}$$

in the 3D Cartesian space. Further, for each point $\mathbf{p} \in \mathcal{P}$, a line-of-sight $\mathbf{s}$, an expected deviation σ, and a reference sample density δ, as described in Chapter 2, exist.

Point Neighborhood

Let $\mathcal{N}_R(\mathbf{q}, \mathcal{P})$ denote the **ball point neighborhood** or **euclidean point neighborhood** of a query point $\mathbf{q} \in \mathbb{R}^3$ within a point set $\mathcal{P}$

$$\mathcal{N}_R(\mathbf{q}, \mathcal{P}) := \{\mathbf{p} \in \mathcal{P} \mid d_p^2(\mathbf{q}, \mathbf{p}) \leq R^2\} \tag{3.1}$$

with the neighborhood radius R. The function $d_p(\mathbf{q}, \mathbf{p})$ is the unsigned distance between two points and is further defined in Equation (A.1) in Appendix A. A similar definition of a ball neighborhood is presented in the work of Floater [FR01]. Point $\mathbf{q}$ is not necessarily part of $\mathcal{P}$. If $\mathbf{q}$ is part of $\mathcal{P}$, it is also included in its own neighborhood $\mathcal{N}_R(\mathbf{q}, \mathcal{P})$.

Further, a subset $\mathcal{N}_{R,k}(\mathbf{q}, \mathcal{P}) \subseteq \mathcal{P}$ is called **k-in-R point neighborhood** of $\mathbf{q}$, if and only if it satisfies

$$\begin{aligned} &|\mathcal{N}_{R,k}(\mathbf{q}, \mathcal{P})| = \min\{k, \mathcal{N}_R(\mathbf{q}, \mathcal{P})\} \\ &\forall \mathbf{p} \in \mathcal{N}_{R,k}(\mathbf{q}, \mathcal{P}) \, \forall \tilde{\mathbf{p}} \in \mathcal{N}_R(\mathbf{q}, \mathcal{P}) \setminus \mathcal{N}_{R,k}(\mathbf{q}, \mathcal{P}) \; : \; d_p(\mathbf{p}, \mathbf{q}) \leq d_p(\tilde{\mathbf{p}}, \mathbf{q}) \end{aligned} \tag{3.2}$$

3.1.2 Triangle Meshes

Edges and Triangles

Let $\overline{\mathbf{ab}}$ denote the line segment or **edge** connecting the two points $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ with $\mathbf{a} \neq \mathbf{b}$

$$\overline{\mathbf{ab}} := \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{x} = \lambda_1 \mathbf{a} + \lambda_2 \mathbf{b}; \lambda_1 + \lambda_2 = 1; \lambda_1, \lambda_2 \geq 0\} \; .$$

Analogously, let $\Delta(\mathbf{a}, \mathbf{b}, \mathbf{c})$ denote the **triangular face** connecting the non-collinear points $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$

$$\Delta(\mathbf{a}, \mathbf{b}, \mathbf{c}) := \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{x} = \lambda_1 \mathbf{a} + \lambda_2 \mathbf{b} + \lambda_3 \mathbf{c}; \lambda_1 + \lambda_2 + \lambda_3 = 1; \lambda_1, \lambda_2, \lambda_3 \geq 0\} \; .$$

Mesh Definition

A **triangle mesh** $\mathcal{M}$ is a piecewise, linear approximation of an unknown surface S by triangular faces and is described by the triple

$$\mathcal{M} := (\mathcal{V}_\mathcal{M}, \mathcal{E}_\mathcal{M}, \mathcal{T}_\mathcal{M}) \ ,$$

composed of a set of n vertices

$$\mathcal{V}_\mathcal{M} := \{\mathbf{v}_1, \dots \mathbf{v}_n\} \in \mathbb{R}^3 \ ,$$

a set of m edges

$$\mathcal{E}_\mathcal{M} := \{e_1, \dots, e_m\} \ ,$$

and a set of k triangles

$$\mathcal{T}_\mathcal{M} := \{t_1, \dots, t_k\} \ ,$$

satisfying

$$\forall e \in \mathcal{E}_\mathcal{M} \ : \ \exists \mathbf{a}, \mathbf{b} \in \mathcal{V}_\mathcal{M} \ : \ e = \overline{\mathbf{ab}}$$

and

$$\forall t \in \mathcal{T}_\mathcal{M} \ : \ \exists \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{V}_\mathcal{M} \ : \ t = \Delta(\mathbf{a}, \mathbf{b}, \mathbf{c}) \wedge \overline{\mathbf{ab}}, \overline{\mathbf{ac}}, \overline{\mathbf{bc}} \in \mathcal{E}_\mathcal{M} \ .$$

The vertices in $\mathcal{V}_\mathcal{M}$ are the supporting points of the triangular faces, defined by their coordinates. Here, for every vertex $\mathbf{v} \in \mathcal{V}_\mathcal{M}$ a corresponding surface normal $\mathbf{n}$ rather than the per-point attributes of a sample point exists.
An edge $e \in \mathcal{E}_\mathcal{M}$ is the line segment connecting two adjacent vertices along the surface S. The set of edges may not contain duplicated edges and edges only intersect in their end points. Hence, the set of edges $\mathcal{E}_\mathcal{M}$ must further satisfy

$$\begin{aligned} &\forall i, j \in \{1, \dots, m\} \ : \ i \neq j \Rightarrow e_i \neq e_j \\ &\forall e, \tilde{e} \in \mathcal{E}_\mathcal{M} \ : \ e \cap \tilde{e} \in \mathcal{V}_\mathcal{M} \cup \{\} \end{aligned} \ .$$

A triangle face $t \in \mathcal{T}_\mathcal{M}$ connects its defining edges and the corresponding vertices. Analogous to the set of edges, the set of triangles must not contain duplicates. Further, two triangles can only intersect by their edges or vertices. Hence, $\mathcal{T}_\mathcal{M}$ must further satisfy

$$\begin{aligned} &\forall i, j \in \{1, \dots, k\} \ : \ i \neq j \Rightarrow t_i \neq t_j \\ &\forall t, \tilde{t} \in \mathcal{T}_\mathcal{M} \ : \ t \cap \tilde{t} \in \mathcal{E}_\mathcal{M} \cup \{\} \end{aligned} \ .$$

Point and Edge Neighborhood

The **vertex ball neighborhood** of a vertex $\mathbf{q}$ in the mesh $\mathcal{M}$ is the set

$$\mathcal{N}_R^\mathcal{V}(\mathbf{q}, \mathcal{M}) := \{\mathbf{v} \in \mathcal{V}_\mathcal{M} \mid d_p^2(\mathbf{q}, \mathbf{v}) \leq R^2\} \tag{3.3}$$

with the neighborhood radius R and the unsigned distance $d_p(\mathbf{q}, \mathbf{v})$, similar to the definition of the ball neighborhood in a sample point set, described in Equation (3.1).

Let $\mathrm{d}_e(e, \mathbf{p})$ denote the shortest unsigned distance between the edge e and the point $\mathbf{p}$, as described in Appendix A.2. Hence, the set

$$\mathcal{E}_R(\mathbf{q}, \mathcal{M}) := \{e \in \mathcal{E}_\mathcal{M} \mid d_e^2(e, \mathbf{q}) \leq R^2\} \tag{3.4}$$

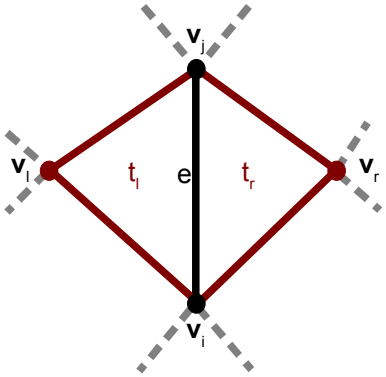


Figure 3.2: Edge data structure in a triangle mesh: An edge data structure consists of two vertices defining the line segment (black), and two additional vertices closing the adjacent triangles (red) to the left and right.

contains all edges of $\mathcal{M}$ that are at least partially inside a ball neighborhood with radius R around $\mathbf{q}$. Further, the set

$$\mathcal{V}(\mathcal{E}_R(\mathbf{q}, \mathcal{M})) := \{\mathbf{v} \in \mathcal{V}_\mathcal{M} \mid \exists e \in \mathcal{E}_R(\mathbf{q}, \mathcal{M}) \ : \ \mathbf{v} \in e\}$$

denotes the corresponding set of vertices in the edge neighborhood. The set of vertices is not equal to the neighborhood $\mathcal{N}_R^\mathcal{V}(\mathbf{q}, \mathcal{M})$, since not every vertex has corresponding edges and not both vertices of an edge $e \in \mathcal{E}_\mathcal{N}$ are inside the spherical volume. Consequently, the **edge ball neighborhood** with radius R of a query point $\mathbf{q}$ in $\mathcal{M}$ is the pair

$$\mathcal{N}_R^\mathcal{E}(\mathbf{q}, \mathcal{M}) := (\mathcal{V}(\mathcal{E}_R(\mathbf{q}, \mathcal{M})), \mathcal{E}_R(\mathbf{q}, \mathcal{M})) \tag{3.5}$$

3.1.3 Representation of a Triangle Mesh

A data structure representing a mesh must allow a local movement through the mesh i.e. stepping from a starting vertex, edge, or triangle to adjacent vertices, edges, and triangles. The central element representing connectivity in a triangle mesh is the edge, because it connects adjacent vertices and triangles along the surface. Common edge structures for general polygonal meshes, such as the winged edge, quad-edge, or half-edge structure, are summarized in the textbook of O'Rourke [O'R98]. These structures are designed for handling arbitrary types of meshes, consisting of polygons with varying numbers of vertices. Thus, they require additional memory per edge and computational effort when modifying the mesh. In this work, a data structure that requires only little memory but is specialized to triangle meshes is used:
Every edge $e \in \mathcal{E}_\mathcal{M}$ is represented by the data structure illustrated in Fig. 3.2. It consists of a starting vertex $\mathbf{v}_i$ and an end vertex $\mathbf{v}_j$, as well as a left vertex $\mathbf{v}_l$ and a right vertex $\mathbf{v}_r$, defining the left and right adjacent triangle. The adjacent faces are fully described by $t_l = \Delta(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_l)$ and $t_r = \Delta(\mathbf{v}_j, \mathbf{v}_i, \mathbf{v}_r)$, because only triangles are allowed as face type. Beside this edge structure, the set of edges

$$\mathcal{E}(\mathbf{v}) = \{e \in \mathcal{E}_\mathcal{M} \mid \mathbf{v} \in e\}$$

is stored for every vertex $\mathbf{v} \in \mathcal{V}_\mathcal{M}$.
Starting from an edge $e \in \mathcal{E}_\mathcal{M}$, the adjacent triangles and vertices can be reached by the edge structure. Starting from a vertex $\mathbf{v} \in \mathcal{V}_\mathcal{M}$, the edges starting or ending in it can be accessed by the attached set $\mathcal{E}(\mathbf{v})$. The adjacent triangles are accessible via the edges in $\mathcal{E}(\mathbf{v})$.

Consequently, the triangles are fully described by their corresponding edges and it is not necessary to store them explicitly. Hence, a mesh storage only contains vertices and edge elements. The design of a mesh storage with the introduced description for vertices and edges is further discussed in Chapter 4.

3.2 Density Limitation

During the process of manually scanning an object, multiple overlapping sweeps with the scanner can generate spots of high local point density in the accumulated point set $\mathcal{P}$. This generates redundancies in the data and increases the overall calculation effort without improving the result. Moreover, assumptions concerning the sample density in subsequent processing stages are violated. For this reason, the density of the accumulated sample points is limited before any further processing.
In this work, a limitation is applied to the incoming streams of both principal stages, the normal estimation and the mesh generation, as illustrated in Fig. 3.1 on page 29. It is coupled to the process of inserting new points into the point set $\mathcal{P}$, since local density has to be verified prior to or during the insertion of the point. The design of a point data storage enabling fast access to subsets of the data is the topic of Chapter 4. In the following, two methods for a streaming density limitation, performed at the insertion of new points into the accumulated point set, are delineated.

3.2.1 Simple Limitation

The simple limitation method rejects points that are located too close to a point that is already contained in the point set, as they would add undesired high sample densities in the point set. This approach was first introduced by Bodenmüller [BH04] and can be summarized by the following rule:

Definition 3.1 *A new point $\mathbf{q} \notin \mathcal{P}$ is inserted into the accumulated point set $\mathcal{P}$ and passed to subsequent processing, iff no point $\mathbf{p} \in \mathcal{P}$ is closer to $\mathbf{q}$ than a limitation radius $R_{\min}$.*

This rule can be rewritten using the ball neighborhood definition from Equation (3.1):

Definition 3.2 *A new point $\mathbf{q} \notin \mathcal{P}$ is inserted into the accumulated point set $\mathcal{P}$ and passed to subsequent processing, iff $\mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}) = \{\}$.*

The simple limitation can be visualized as a set of solid balls with radius $\frac{R_{\min}}{2}$ that can not penetrate each other, one for each point in $\mathcal{P}$. This is illustrated for two dimensions in Fig. 3.3. The maximum number of balls in a finite volume V can be derived from the average density in a close-packing of spheres. The highest average density in such an arrangement is

$$\frac{nV_{Sp}}{V} = \frac{\pi}{3\sqrt{2}} \ , \tag{3.6}$$

with the number of spheres n and the volume of a sphere V_{Sp}[1]. The substitution of $V_{Sp} = \frac{1}{6}\pi R_{\min}^3$ in Equation (3.6) results in the maximum number of points $n_{V_{\max}}$, limited

[1]The formula denotes the highest average density of a regular lattice arrangement. The Kepler conjecture states that this is also the highest density that can be achieved by any arrangement of spheres, either regular or irregular. A proof has been given in the work of Thomas Hales [Hal05].

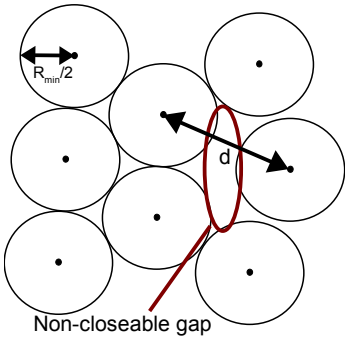


Figure 3.3: Gaps between points in a limited point set: A point with a distance d with $R_{\min} < d < 2R_{\min}$ to the adjacent point decreases the local point density w.r.t. the close packing of spheres due to the larger gap between the points.

by the radius $R_{\min}$, in an arbitrary finite volume V

$$n_{V_{\max}} = \frac{\sqrt{2}\,V}{R_{\min}^3} \ . \tag{3.7}$$

Concluding, the number of points in a finite volume can not be arbitrarily high, but is coupled to the size of the volume and the limitation radius.
However, the sample points are inserted in random order and may have a distance larger than $R_{\min}$ to the nearest point in the point set. Hence, the density limitation potentially creates larger non-closable gaps between the balls. The result is a local point density that is lower than that of a pure close packing of spheres would generate. As an example, two points $\mathbf{p}_1$ and $\mathbf{p}_2$ with a euclidean distance $d = \|\mathbf{p}_2 - \mathbf{p}_1\|$ that are consecutively inserted into the point set result in a non-closable gap, if the distance d is in the range

$$R_{\min} < d < 2R_{\min} \ ,$$

i.e. no other point can be inserted in between. This is further illustrated in Fig. 3.3. In the worst case, all points in the limited point set have a distance to each other of $d = 2R_{\min}$, resulting in a sphere arrangement of balls with radius $R_{\min}$. Thus, a pessimistic upper bound to the number of points $n_{V_{0.5\,\max}}$ for a finite volume V is

$$n_{V_{0.5\,\max}} = \frac{V}{4\sqrt{2}R_{\min}^3} \ . \tag{3.8}$$

3.2.2 Limitation with Replacement

The simple limitation approach generates an undesired dependency on the input order of the sample points. Points with a higher quality can be rejected, because points of lower quality have been inserted earlier within a distance of $R_{\min}$. This reflects the assumption that a comparable quality of each sample point exists and not all points are of the same quality. However, a simple replacement of an old point with a new one with higher quality would lead to a stepwise shift of the points, i.e. the replacement implicates that the point's coordinate changes and thus possibly generates a gap that is large enough for an other point. This shift could cause violation of the limitation rule, invalidating the upper bound for the number of points given by Equation (3.7). Hence, an replacement approach is required that account for point quality but avoids that coordinate changes can violate the density limitation.

In this work, the problem is tackled by adding an separate anchor point $\mathbf{p}_a$ to each sample point $\mathbf{p}$. The sample point $\mathbf{p}$ can be replaced while the anchor point $\mathbf{p}_a$ remains unchanged and is used for the density limitation. Hence, a set of anchor points $\mathcal{P}_a$ is required beside the sample point set $\mathcal{P}$ and for each sample point $\mathbf{p} \in \mathcal{P}$ there exists an anchor point $\mathbf{p}_a \in \mathcal{P}_a$. This replacement limitation method is pictured in Alg. 1 and is detailed in the following.
Let f_q denote the function

$$f_q : \mathbb{R}^3 \to \mathbb{R} \; ; \; \mathbf{p} \mapsto q$$

that maps $\mathbf{x}$ to a scalar quality value $q_{\mathbf{x}}$. A new point $\mathbf{q}$ from stream is tested according to Definition 3.2, however, in relation to a set of anchor points $\mathcal{P}_a$ rather than the point set $\mathcal{P}$. If the new point passes the simple limitation with the anchor point set $\mathcal{P}_a$, it is inserted into the sets $\mathcal{P}$ and $\mathcal{P}_a$, i.e. it defines a new anchor point. Otherwise, the quality q of the point $\mathbf{q}$ is compared to the quality of all points $\mathbf{p} \in \mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}_a)$. The new point replaces the nearest point in $\mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}_a)$ that has a lower quality than q (if any), while the corresponding anchor point is not modified. Hence, the coordinate of an already inserted point can only change within the sphere volume centered at the corresponding anchor point and with the radius $R_{\min}$. In this work, the corresponding expected deviation σ of a point $\mathbf{p}$, as introduced in Chapter 2.4, is applied as quality criterion

$$q = f_q(\mathbf{p}) := \sigma \; .$$

Alternatively, other attributes, e.g. sample density or intensity, can be used.

Algorithm 1 Replacement density limitation at the insertion of point $\mathbf{q}$.

$\mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}_a) \leftarrow \mathcal{P}_a$ /* Query of $\mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}_a)$ */
if $\mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}_a) = \{\}$ **then**
  /* Point passes the limitation stage : insert into point set */
  $\mathcal{P} := \mathcal{P} \cup \mathbf{q}$
  $\mathcal{P}_a := \mathcal{P}_a \cup \mathbf{q}$
else
  /* Point is rejected : find nearest point with lower quality */
  $d_{\min} := R_{\min}$ /* distance of nearest point */
  $*\mathbf{p}_{\min} := 0$ /* Nearest point */
  for all $\mathbf{p}_a \in \mathcal{N}_{R_{\min}}(\mathbf{q}, \mathcal{P}_a)$ **do**
    $\mathbf{p}, f_q(\mathbf{p}) \leftarrow \mathbf{p}_a$ /* Get corresponding point and quality to anchor point */
    if $\|\mathbf{p}_a - \mathbf{q}\| < d_{\min}$ **and** $f_q(\mathbf{p}) < f_q(\mathbf{q})$ **then**
      $d_{\min} := \|\mathbf{p}_a - \mathbf{q}\|$
      $\mathbf{p}_{\min} := \mathbf{p}$
    end if
  end for
  if $\mathbf{p}_{\min} \neq 0$ **then**
    /* case 1 : replacement */
    $\mathbf{p}_{\min} := \mathbf{q}$
    $f_q(\mathbf{p}_{\min}) := f_q(\mathbf{q})$
  else
    /* case 2 : rejection */
  end if
end if

This extended limitation method decreases the dependency upon the input order, compared to the simple approach. Initially, all points are used until the maximum

density is reached. Any point is used as long as no point with higher quality is available. Consequently, adding more accurate points to a noisy point set leads to an incremental improvement of the set's accuracy instead of rejection of these higher quality points. Hence, the operator of a manual scanning system can optimize the results by performing more scans.

3.3 Estimation of Surface Normals

The surface normal for each sample point is the first local surface property that can be derived from an unorganized point set. It represents the tangent plane to the surface at the respective point. Consequently, a surface normal is mandatory for most local surface approximation techniques.
The estimation of the surface normals for the sample points of an unorganized point set is generally solved by least-square fitting of a tangent plane through a point neighborhood of the examined point. In the early work of Hoppe [Hop94], a covariance matrix is calculated from the k-nearest neighbors. The eigenvectors are derived, representing the least-square tangent plane. Contrary, in the work of Gopi [Gop01] the eigenvectors from a $k \times 3$ matrix of differences are calculated, with the number of neighboring points k, using SVD[2], delivering similar results as the covariance approach. A more complex method for fitting a tangent plane to a local point set is the first stage of the projection procedure presented in the work of Alexa et al. [ABCO$^+$03] it is used for the construction of a MLS[3] surface. Here, a non-linear least square optimization is performed for fitting the plane.
Many approaches assume a global homogeneous and constant noise and density. However, it is difficult to find a scan path which ensures a homogeneous digitization of a non-trivial surface. For manual scanning, the local sampling density and noise vary depending on the scan path and scan speed applied by the operator, as well as the measured depth itself, as discussed in Chapter 2. Moreover, multiple sweeps conducted for specific regions generate a high local density, whereas other regions may only be sparsely sampled, as discussed in the previous Chapter 3.2. The variation of noise and density is even higher, if data from different sensors with different characteristics are integrated into a single 3D model.
Therefore, the method presented in the following accounts for variable sample density and noise. Moreover, the streaming character of the input data is considered, resulting in a method without global operations regarding the point set, the query of a local point neighborhood being the only exception.

3.3.1 Basic Estimation Method

The surface normal estimation method used is based on the covariance approach applied in the work of Hoppe [Hop94] or Pauly et al. [PGK02]. The method is simple, thus the calculation can be performed rapidly. Despite its simplicity, the method is well suited for normal estimation, even if two opposing surfaces are located closely together. The original approach is extended, so that a local neighborhood of potential inhomogeneous noisy points can be regarded.
Let $\mathbf{q}$ denote the examined point and let $\mathbf{n}$ be the surface normal to be estimated at $\mathbf{q}$. Further, let $\mathcal{P}_{\mathcal{N}}(\mathbf{q}) = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subseteq \mathcal{P}$ denote a local neighborhood around $\mathbf{q}$ that

[2]SVD = Singular Value Decomposition
[3]MLS - Moving Least Squares

represents the local shape and contains points that are sufficiently distributed. For every point $\mathbf{p}_i \in \mathcal{P}_\mathcal{N}(\mathbf{q})$, a normalized scalar weight

$$w_i := \frac{\sigma_i^{-1}}{\sum_{i=1}^{n} \sigma_i^{-1}}$$

with the expected deviation σ_i corresponding to $\mathbf{p}_i$ exists. Further, the weighted covariance matrix **Cov** is the 3×3-matrix

$$\mathbf{Cov} := \frac{1}{1-\bar{w}^2} \sum_{i=1}^{n} w_i (\mathbf{p}_i - \mathbf{c})(\mathbf{p}_i - \mathbf{c})' \quad \mathbf{p}_i \in \mathcal{P}_\mathcal{N}(\mathbf{q}) \tag{3.9}$$

with the mean

$$\mathbf{c} := \sum_{i=1}^{n} w_i \mathbf{p}_i \tag{3.10}$$

and the squared sum of weights

$$\bar{w}^2 := \sum_{i=1}^{n} w_i^2 \, .$$

The mean and the covariance matrix are unbiased estimates of the true mean and covariance matrix[4]. For a constant deviation $\sigma_i = \sigma$, the scalar weight is reduced to $w = \frac{1}{n}$ and the covariance calculation is reduced to the standard equation presented in the work of Pauly et al. [PGK02]

$$\mathbf{Cov} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{p}_i - \mathbf{c})(\mathbf{p}_i - \mathbf{c})' \quad \mathbf{c} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{p}_i \, .$$

Let $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ denote the eigenvalues of **Cov** and let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ denote the corresponding eigenvectors[5]. According to the *principal component analysis (PCA)*, the eigenvectors are the principal axes of the set $\mathcal{P}_\mathcal{N}$. The first axis $\mathbf{v}_1$ is the direction of maximum deviation and $\mathbf{v}_2$ is the axis of maximum deviation perpendicular to $\mathbf{v}_1$. The vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ span the least-square tangent plane through $\mathcal{P}_\mathcal{N}$. The axis $\mathbf{v}_3$ is the normal vector on this plane and thus an estimate for the surface normal in $\mathbf{q}$, assuming that the point neighborhood has a significantly higher deviation along the surface than perpendicular to it.
However, both directions, $\mathbf{v}_3$ and $-\mathbf{v}_3$, are possible estimates for the surface normal. In the work of Pauly et al. [PGK02], a minimal spanning tree method is used to propagate a consistent orientation of the surface normals in a post processing step. In this work, the surface normal estimation is applied to points generated by a real scanner system. Hence, the line-of-sight $\mathbf{s}$ of the examined point $\mathbf{q}$ is used to distinguish between the inside and the outside of the scanned object, and thus assures a consistent orientation of the surface normals. Finally, the estimated surface normal $\mathbf{n}$ at the examined point $\mathbf{q}$ is

$$\mathbf{n} = \mathbf{v}_3 \cdot \text{sign}(\langle \mathbf{v}_3, -\mathbf{s} \rangle) \, . \tag{3.11}$$

The method only provides correct estimations, if the point neighborhood consists of a sufficient number of points, is homogeneously distributed around $\mathbf{q}$, and represents the local shape of the surface. As shown in the work of Mitra and Nguyen [MN03],

[4] See *GNU Scientific Library Reference manual, Vers. 1.9, 2007, Sec. 20.6 Weighted Samples* for details.
[5] The covariance matrix is a real symmetric positive definite matrix. The eigenvalues and eigenvectors can be calculated very fast using the **jacobi method** (see [PTVF92]).

noise and curvature can result in wrong estimation results, if the neighborhood is not chosen properly. Moreover, the streaming character of the process has to be considered. Therefore, the following sections cover the incremental update of surface normals and the calculation of a suitable point neighborhood. The verification of estimation results is discussed in Section 3.4.

3.3.2 Incremental Update

The use of local point neighborhoods, such as k-nearest neighbors or ball neighborhoods for the surface normal estimation implicates that a newly inserted point influences its local neighborhood and vice versa. Hence, the use of an estimation algorithm in a processing pipeline for streaming point data requires a continuous update of the surface normals of all influenced points.
A new point $\mathbf{q}$ that has passed the density limitation stage potentially influences the surface normals of all points in its neighborhood $\mathcal{P}_\mathcal{N}(\mathbf{q})$. Consequently, the corresponding surface normals of all points in the set $\mathcal{P}_\mathcal{N}(\mathbf{q}) \cup \mathbf{q}$ have to be re-estimated. This procedure is visualized in Alg. 2: First, the point neighborhood $\mathcal{P}_\mathcal{N}(\mathbf{q})$ is determined. For every point $\mathbf{p} \in \mathcal{P}_\mathcal{N}(\mathbf{q})$ its own point neighborhood $\mathcal{P}_\mathcal{N}(\mathbf{p})$ is updated and the surface normal is re-estimated. A smoothed point $\bar{\mathbf{p}}$ is calculated, using the estimated mean $\mathbf{c}$ (see Equation (3.10)). In order to avoid a shift of the point along the surface caused by an unbalanced neighborhood, only an averaging in normal direction is applied by

$$\bar{\mathbf{p}} = \mathbf{p} + \langle (\mathbf{c} - \mathbf{p}), \mathbf{n} \rangle \ \mathbf{n} \quad . \tag{3.12}$$

The use of the weighted center $\mathbf{c}$ results in a non-homogeneous smoothing w.r.t. the deviation of the respective point. Thus, noisy points are smoothed more than accurate ones. Finally, each pair $(\bar{\mathbf{p}}, \mathbf{n})$, consisting of the smoothed point and the estimated normal, is transfered to the selection stage for verification.

Algorithm 2 Incremental update of the surface normals at the insertion of a new point $\mathbf{q}$.

```
Calculate P_N(q) /* Section 3.3.3 */
for all p ∈ P_N(q) ∪ q do
  Update neighborhood P_N(p) /* Section 3.3.3 */
  if P_N(p) has changed then
    Re-estimate surface normal n /* Section 3.3.1 */
    Calculate smoothed point p̄ /* Equation (3.12) */
    Transfer the pair (p̄, n) to selection stage /* Section 3.4 */
  end if
end for
```

A single input point $\mathbf{q}$ generates a set of potential output points $\mathcal{P}_\mathcal{N}(\mathbf{q})$. Depending on the used neighborhood strategy applied, not all surface normals are updated necessarily. This fact is discussed in the next section.

3.3.3 Choice of Point Neighborhood

The correctness of surface normals depends on the set of local neighboring points $\mathcal{P}_\mathcal{N}$ used for their estimation. In the work of Hoppe [Hop94], the k-nearest neighbors are used. Contrary, in the work of Bodenmüller [BH04], a ball neighborhood with constant radius, similar to Equation (3.1), is applied. The key advantage of the k-nearest

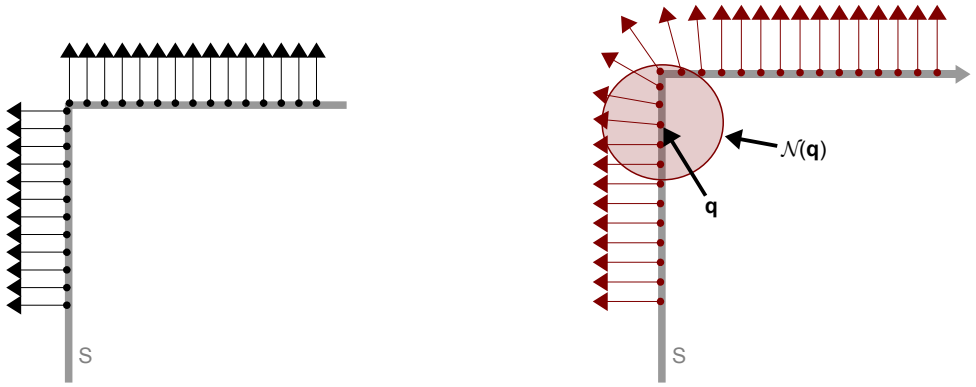


Figure 3.4: Normal estimation at sharp edges: The original surface normals (left) and the estimated normals (right).

neighbors is the constant number of points in the neighborhood, which limits the computational effort of the surface normal estimation. However, the k-nearest neighbors typically do not represent a defined area on the surface at an inhomogeneous point density. A ball neighborhood, however, is a volume set around the examined point and thus defines a finite area on the surface. It can consist of an arbitrary number of points. Consequently, this work combines the concepts for point sets with density limitation (see Section 3.2), enabling a continuous optimization of the neighborhood for each point individually.
Generally, a robust normal estimation requires that the covered surface area of the neighborhood $\mathcal{P}_\mathcal{N}$ is sufficiently large w.r.t. local sample density and noise of the respective point. A neighborhood that is too small can result in an insufficient number of points for estimation. Moreover, sensor noise and high curvature can lead to bad estimation results, because the deviation of points in normal direction is not significantly smaller then the deviation along the surface.
However, the computational effort increases with the number of points in the neighborhood. Moreover, the local shape of the surface is not represented, if the covered area is too large. The estimation of surface normals from a local neighborhood always causes a continuous change of normal direction, even at a sharp edge. This fact is illustrated in Fig. 3.4 for a ball neighborhood $\mathcal{N}_R(\mathbf{q})$. The smaller the neighborhood radius, the faster the normals change along the curved surface. A radius chosen too large can result in a loss of local shape details and thus causes problems in the subsequent triangulation stage.
A key goal for manual scanning processes and surface reconstruction for visual feedback is to generate valid surface normals as quickly as possible. However, the neighborhood information of an examined point can be sparse in the beginning, growing only with the insertion of new points from the stream. Consequently, a neighborhood initially has to cover a large area, so a coarse surface normal can be estimated and instantly be used for reconstruction and visualization. This surface normal should reflect the local shape with the computational effort for estimation should be kept low. This requires a neighborhood that has a limited number of points and only covers the area that is needed for robust estimation.
In this work, the **k-in-R point neighborhood** is used to meet these requirements:

$$\mathcal{P}_\mathcal{N}(\mathbf{q}) := \mathcal{N}_{R_n,k_n}(\mathbf{q},\mathcal{P}) \ .$$

Every newly inserted point $\mathbf{q}$ starts with an initial ball neighborhood radius $R_n := R_{n0}$ but only the k_n-nearest neighbors of the ball neighborhood are used, as described in

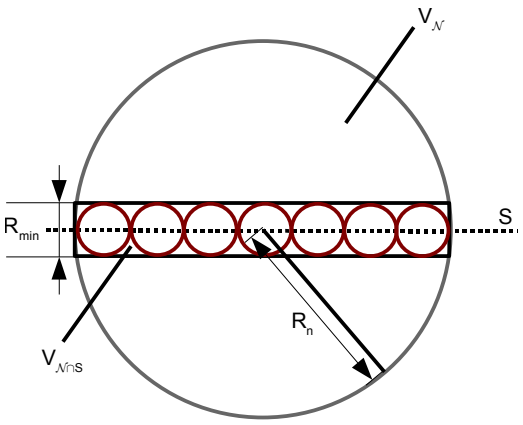


Figure 3.5: Calculation of the minimum neighborhood radius: The use of the ball neighborhood volume $V_{\mathcal{N}}$ results in a bad guess for the maximum number of sample points, since the points can not have arbitrary locations in the volume but are located on the unknown surface S. Contrary, the cylindrical volume $V_{\mathcal{N}\cap S}$ covers only the area along the surface w.r.t. the limitation radius $R_{\min}$. This results in a more realistic guess for the maximum nuber of sample points.

Equation (3.2), potentially shrinking the neighborhood with every update.
Hence, the insertion of a new point $\mathbf{p}$ that is nearer than the farthest neighboring point results in a successive decreasing of the neighborhood radius

$$\mathcal{N}_{R_n^*,k_n}(\mathbf{q},\mathcal{P}) = \mathcal{N}_{R_n(\mathbf{q}),k_n}(\mathbf{q},\mathcal{P}\cup\{\mathbf{p}\}) \ ,$$

with the new ball neighborhood radius

$$R_n^* = \max\{d_p(\mathbf{p}_i,\mathbf{q}) \in \mathbb{R} \mid \mathbf{p}_i \in \mathcal{N}_{R_n,k_n}(\mathbf{q},\mathcal{P})\cup\{\mathbf{p}\}\} \ .$$

However, the neighborhood radius R_n can not become arbitrarily small w.r.t. the density limitation discussed in Section 3.2. The maximum number of points in a finite volume depends on the limitation radius $R_{\min}$, as shown in Equation (3.7).
Let $R_{\min}$ denote the limitation radius and let k_n be the number of points in the neighborhood. The volume of the neighborhood sphere with radius R_n is

$$V_{\mathcal{N}} = \frac{4}{3}\pi R_n^3 \ .$$

The substitution of the finite volume V in Equation (3.7) with the neighborhood volume $V_{\mathcal{N}}$ and with $n_{V_{\max}} = k_n$ results in a minimum neighborhood radius of

$$R_{n_{\mathcal{N}}}^3 = \frac{3k_n R_{\min}^3}{4\sqrt{2}\pi} \ .$$

However, this bound is a bad guess w.r.t. the sampled surface S, because the sample points are on the surface and can not have arbitrary locations inside the neighborhood sphere. A better approximation for a minimum neighborhood radius is found by restricting the spherical volume w.r.t. a planar reference surface. This is approximated by a cylindrical volume

$$V_{\mathcal{N}\cap S} = \pi R_n^2 R_{\min} \ ,$$

as illustrated in Fig. 3.5. The substitutions $V = V_{\mathcal{N}\cap S}$ and $n_{V_{\max}} = k_n$ in Equation (3.7) result in the neighborhood radius

$$R_{n_{\mathcal{N}\cap S}}^2 = \frac{k_n R_{\min}^2}{\sqrt{2}\pi} \ . \tag{3.13}$$

The same substitutions performed in Equation (3.8) for the worst case density bound implicates a doubled limitation radius. Hence, the cylindrical volume must also be doubled in height. This results in an neighborhood radius

$$R^2_{n_{0.5\mathcal{N}\cap S}} = \frac{4\,k_n\,R^2_{\min}}{\sqrt{2}\pi} \ . \tag{3.14}$$

A minimum neighborhood radius $R_{n_{\min}}$ is thus always linearly coupled to the limitation radius $R_{\min}$ and increases with the square root of k_n. The bounds for $R_{n_{\min}}$ given by Equation (3.13) and Equation (3.14), resulting in

$$R^2_{n_{\min}} = \frac{\rho k_n\,R^2_{\min}}{\sqrt{2}\pi} \qquad \rho \in [1,4] \ .$$

The factor ρ controls the trade-off between computational effort and coverage of the neighborhood. In this work, a factor of $\rho = 2.5$ is used. The minimum neighborhood radius and the per-point attributes $\delta_{\mathbf{q}}$ and $\sigma_{\mathbf{q}}$ of each sample point $\mathbf{q}$ can be applied to control the initial neighborhood radius R_{n0} individually. This aspect is further discussed in Chapter 5.

3.4 Selection and Tracking

In the previous section, a basic surface normal estimation method and the determination of a sufficient point neighborhood for this estimation has been presented. However, the neighborhood of a newly inserted point can still contain too few points or the points in the neighborhood are not evenly distributed, resulting in poor estimation results. Moreover, noise and outliers can cause bad estimates. Hence, the estimated surface normals have to be checked for correctness in order to avoid non-plausible surface normals in successive stages. Points that have already been transmitted can also change in position, due to replacement, or in direction of the normal, due to an update. These changes have to be tracked and conveyed to the successive stage. In the following, criteria for an initial **selection of points** and the further **tracking of changes** are explained.

3.4.1 Initial Selection of Points

Selection denotes the verification of all modified surface normals after a normal update, until the respective normal with its corresponding point enters the subsequent mesh generation stage. During mesh generation, a point with a miss-estimated surface normal would lead to cracks in the generated surface. Hence, it is mandatory to block points with bad surface normals and to transfer only points with correct normals. However, as the true surface normal is not known, the estimated surface normals can only be verified w.r.t. their plausibility, not their correctness.
Both, noise and curvature can lead to inaccurate normal estimations, as remarked in the work of Mitra and Nguyen [MN03]. Due to the sight-dependent decision in Equation (3.11), a sample point that has a very flat line-of-sight $\mathbf{s}$ onto the surface and an estimated eigenvector $\mathbf{v}_3$ that is slightly rotated due to noise or curvature can result in an inconsistent or flipped normal direction. This problem of surface normals that are accidentally pointing inside the object is further illustrated in Fig. 3.6. The grazing angle α_s between line-of-sight $\mathbf{s}$ and estimated normal $\mathbf{n}$ should be small enough to

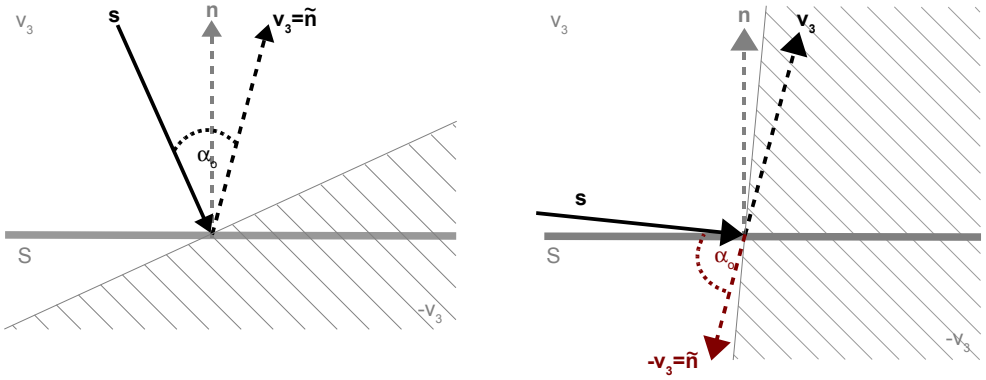


Figure 3.6: Flipped surface normal due to noise and flat line-of-sight: If the line-of-sight is upright on the surface, the sign of the estimated surface normal can be robustly determined, even if v_3 is noisy (a). If the line-of-sight is flat, the surface normal can flip due to the noisy estimation v_3 i.e. it is pointing inside the object (b).

guarantee a consistent direction of the surface normal $\mathbf{n}$. This is verified with the criterion $\mathfrak{c}_0$ that requires the grazing angle α_s to be smaller than a threshold $\alpha_{s_{\max}}$,

$$\mathfrak{c}_0 : \alpha_s = \arccos\left(\langle \mathbf{n}, -\mathbf{s} \rangle\right) < \alpha_{s_{\max}} \ . \tag{3.15}$$

Hence, the criterion $\mathfrak{c}_0$ verifies the measurability of a sample point w.r.t. the line-of-sight of the sensor.

For further verification, the deviation of the point neighborhood has to be analyzed. As explained in Section 3.3, the normal estimation bases on a principal component analysis. The eigenvectors of the covariance matrix are the principal axes and thus represent the directions of maximum deviation. The corresponding eigenvalues are the variances in each direction with

$$\sigma_i^2 = \lambda_i \ .$$

For the normal estimation in Section 3.3.1 it is assumed that the variances σ_1^2 and σ_2^2 along the surface are significantly higher than the variance σ_3^2, perpendicular to the surface. However, this assumption can be violated by a sparse or unbalanced neighborhood in combination with noise and high curvature. Consequently, it is required that the variances in both directions of the tangent plane are significant higher than the variance in normal direction i.e.

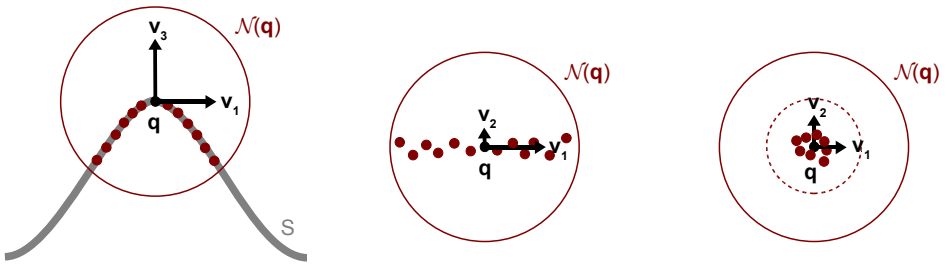
$$\sigma_3^2 << \sigma_2^2 < \sigma_1^2 \ ,$$

in order to achieve a robust estimation result. This is verified by the first variance criterion $\mathfrak{c}_1$ with

$$\mathfrak{c}_1 : \frac{\sigma_3^2}{\sigma_2^2} < 0.5 \ . \tag{3.16}$$

Further, it is required that the distribution of the neighborhood around the considered point is homogeneous. An inhomogeneous neighborhood, e.g. the degenerated case of all points on a single stripe, can result in a $90°$-rotated surface normal. This aspect is validated using two criteria. First, a homogeneous distribution in both tangent plane directions is required

$$\sigma_1^2 \approx \sigma_2^2$$



(a) $\mathfrak{c}_1$ violation (b) $\mathfrak{c}_2$ violation (c) $\mathfrak{c}_3$ violation

Figure 3.7: Violations of the variance criteria: A high curvature w.r.t. the neighborhood size results in a variance σ_3^2 that is not significantly smaller than the other variances, causing a $\mathfrak{c}_1$ violation (a). An example for inhomogeneous neighborhoods is shown in (b). All points are on a stripe, causing a $\mathfrak{c}_2$ violation. The absolute variance is too small, if the neighbors are all very close to the examined point $\mathbf{q}$ (c). This causes a $\mathfrak{c}_3$ violation.

and verified by the criterion $\mathfrak{c}_2$ with

$$\mathfrak{c}_2 : 1 > \frac{\sigma_2^2}{\sigma_1^2} > 0.5 \ . \tag{3.17}$$

Secondly, the absolute variances in the plane directions have to be sufficiently high, validated by criterion $\mathfrak{c}_3$ w.r.t. the neighborhood radius R_n with

$$\mathfrak{c}_3 : \sigma_1^2 + \sigma_2^2 > \left(\frac{R_n}{2}\right)^2 \ . \tag{3.18}$$

Fig. 3.7 shows violations of these variance criteria, one for each of the criteria $\mathfrak{c}_1, \mathfrak{c}_2, \mathfrak{c}_3$.

The criteria for selection are very strict and guarantee a suitable neighborhood for the estimation, however, they delay the data flow. If numerous neighbors of the examined point have already been selected, the estimated surface normal can also be verified by comparison with the normals of these neighboring points. This results in an acceleration of the estimation process, especially in flat surface areas.
Let $\bar{\mathbf{n}}$ denote the average normal of all already selected points in the neighborhood and let n_s denote their number. The examined point will be selected, if n_s is sufficiently high and the angle between surface normal $\mathbf{n}$ and average normal $\bar{\mathbf{n}}$ is small. This is denoted by the criterion $\mathfrak{c}_{fast}$ with

$$\mathfrak{c}_{fast} : \alpha_n = \arccos\left(\langle \mathbf{n}, \bar{\mathbf{n}} \rangle\right) < \alpha_{n_{\max}} \wedge n_s \geq n_{s_{\min}} \ . \tag{3.19}$$

Here, $n_{s_{\min}}$ denotes the minimum number of selected points and $\alpha_{n_{\max}}$ is the maximum directional difference.
In summary, a point will be selected if it fulfills the measurability criterion $\mathfrak{c}_0$ (Equation (3.15)) and either all three variance criteria $\mathfrak{c}_1, \mathfrak{c}_2, \mathfrak{c}_3$ (Equations (3.16)-(3.18)) or the fast-selection criterion $\mathfrak{c}_{fast}$ (Equation (3.19)).

3.4.2 Tracking of Changes

The selection verification is applied until a point passes the tests for the first time, as stated in the previous section. The considered point is marked as selected and along with its corresponding surface normal, is transfered to the subsequent mesh generation stage, as shown in Fig. 3.1.

However, the selection tests merely for plausibility, not for a correct normal. After a point has been selected, its surface normal can still change, due to newly inserted data and the resulting modification of the neighborhood as described in Section 3.3.3. The neighborhood radius potentially decreases and thus represents the local shape more accurately.
Therefore, changes of the surface normals caused by insertion of new points have to be tracked even after a point has been selected. The tracking reports the changes to the subsequent mesh generation stage in which the surface can then be refined. This requires a removal and re-insertion of the changed points into the mesh, increasing the computation effort of the mesh generation process. Hence, only major changes to the surface normal are reported.
Let β denote the angle between the surface normals before update $\tilde{\mathbf{n}}$ and after update $\mathbf{n}$. A point is re-transfered if it fulfills the measurability criterion $\mathfrak{c}_0$ (see Equation (3.15)) and the angle β is larger than a threshold $\beta_{\min}$

$$\beta = \arccos\left(\langle \mathbf{n}, \tilde{\mathbf{n}} \rangle\right) > \beta_{\min} \, .$$

3.5 Localized Triangulation

The mesh generation consists of a limitation and replacement stage followed by a localized triangulation step, as illustrated in Fig. 3.1. A triangle mesh is incrementally built by continuously refining it locally with every newly received point. The triangulation step can be divided to the following processing steps:

1. Limited insertion or replacement
2. Projection and candidate selection
3. Triangulation update
4. Calculation of triangular faces

First, a point received from stream has to pass a limitation step. Then it is stored as vertex of the mesh. Afterwards, the neighboring vertices and edges of the new vertex are projected onto its tangent plane, defined by the surface normal. The subsequent triangulation update is performed in a local 2D space. This general approach is also used in the method of Gopi [Gop01] and for identifying the MLS surface in the work of Alexa et al. [ABCO$^+$03]. In this work, a ball neighborhood is used for identifying a set of candidate points that are used for adding edges to the mesh, denoted as triangulation update. Finally, the set of triangle faces has to be re-calculated, considering possible new insertions and removals of edges. The four stages are detailed in the following sections.

3.5.1 Limited Insertion and Replacement

A point-normal pair $(\bar{\mathbf{p}}, \tilde{\mathbf{n}})$ that passed the normal estimation stage is received by the **Limited Insertion and Replacement** stage. A new point from stream can either originate from an initial selection or from the tracking of major changes, as described in Section 3.4.
Generally, a new point must first pass an additional density limitation step, as described in Section 3.2. Here, the simple limitation pattern is applied, because an optimization of the point set by replacement has already been performed at the initial

limitation in the normal estimation stage. All accepted points are stored as the vertices of the emerging mesh. The limitation guarantees that no two vertices are closer to one another than the limitation radius $R_{e_{\min}}$. Hence, the mesh resolution can be adjusted by this additional limitation step. Consequently, the radius $R_{e_{\min}}$ is called minimum edge length.
A point from an initial selection is inserted as vertex or rejected by the limitation step, as described previously. If the point originates from a tracking of changes, it has already been transfered before. Hence, the corresponding vertex with all adjacent edges and triangles has to be identified and must be removed. Afterwards, the new point is inserted again.

3.5.2 Projection and Candidate Selection

Let $\mathcal{M}$ denote a triangle mesh as defined in Section 3.1 and let $\mathbf{v}$ denote the examined vertex with its corresponding surface normal $\mathbf{n}$. Further, let $\mathcal{N}^{\mathcal{V}}_{R_{e_{\max}}}(\mathbf{v}, \mathcal{M})$ denote the vertex ball neighborhood in $\mathcal{M}$ with radius $R_{e_{\max}}$ around the vertex $\mathbf{v}$. The projection $\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}})$ of a vertex $\tilde{\mathbf{v}}$ onto the tangent plane with origin $\mathbf{v}$ and plane normal $\mathbf{n}$ is the mapping

$$\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}}) : \mathbb{R}^3 \to \mathbb{R}^2; \tilde{\mathbf{v}} \mapsto \tilde{\mathbf{v}}^* \ ,$$

with

$$\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}}) := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \mathbf{R}(\mathbf{n}) \ (\tilde{\mathbf{v}} - \mathbf{v}) \ .$$

The rotation w.r.t. the plane normal $\mathbf{n}$ is

$$\mathbf{R}(\mathbf{n}) := \begin{pmatrix} \mathbf{a} & \mathbf{n} \times \mathbf{a} & \mathbf{n} \end{pmatrix}^T \ ,$$

with the plane direction

$$\mathbf{a} := \begin{cases} \mathbf{n} \times (1\,0\,0)^T & (1\,0\,0) \cdot \mathbf{n} \neq 0 \\ \mathbf{n} \times (0\,1\,0)^T & \text{else} \end{cases} \ .$$

Hence, the set of candidate points $\mathcal{C}(\mathbf{v})$ is the projection of the ball neighborhood $\mathcal{N}^{\mathcal{V}}_{R_{e_{\max}}}(\mathbf{v}, \mathcal{M})$ onto the tangent plane of $\mathbf{v}$ with

$$\mathcal{C}(\mathbf{v}) := \{\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}}) \in \mathbb{R}^2 \mid \tilde{\mathbf{v}} \in \mathcal{N}^{\mathcal{V}}_{R_{e_{\max}}}(\mathbf{v}, \mathcal{M}) \setminus \mathbf{v}\} \ . \tag{3.20}$$

The candidate points are filtered by their surface normals, in order to avoid points from a possible backside surface in the set of candidates. The surface normal of each candidate point must not differ more than a threshold $\alpha_{c_{\max}}$ from the surface normal $\mathbf{n}$ of the examined vertex $\mathbf{v}$. Hence, Equation (3.20) is extended to

$$\mathcal{C}(\mathbf{v}) := \{\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}}) \in \mathbb{R}^2 \mid \tilde{\mathbf{v}} \in \mathcal{N}^{\mathcal{V}}_{R_{e_{\max}}}(\mathbf{v}, \mathcal{M}) \setminus \mathbf{v} \wedge \arccos(\langle \mathbf{n}_i, \mathbf{n} \rangle) < \alpha_{c_{\max}}\} \tag{3.21}$$

with the corresponding surface normal $\tilde{\mathbf{n}}$ of the vertex $\tilde{\mathbf{v}}$.

Further, let $\mathcal{L}(\mathbf{v})$ denote the projection of all edges in the edge ball neighborhood $\mathcal{N}^{\mathcal{E}}_{R_{e_{\max}}}(\mathbf{v}, \mathcal{M})$ onto the tangent plane of $\mathbf{v}$. The neighborhood consists of a set of edges $\mathcal{E}_R(\mathbf{q}, \mathcal{M})$ and a corresponding set of vertices $\mathcal{V}(\mathcal{E}_R(\mathbf{q}, \mathcal{M})$, as defined in Equation (3.5). Only the vertices of the edges have to be projected onto the tangent plane, since the edges $e \in \mathcal{E}_R(\mathbf{q}, \mathcal{M})$ merely represent the connectivity of the vertices. Hence, the set of projected vertices is

$$\mathcal{V}^*(\mathcal{E}_R(\mathbf{q}, \mathcal{M})) := \{\mathrm{Pr}_{\mathbf{v},\mathbf{n}}(\tilde{\mathbf{v}}) \in \mathbb{R}^2 \mid \tilde{\mathbf{v}} \in \mathcal{V}(\mathcal{E}_R(\mathbf{q}, \mathcal{M})) \wedge \arccos(\langle \tilde{\mathbf{n}}, \mathbf{n} \rangle) < \alpha_{c_{\max}}\} \ ,$$

and the projection of the neighborhood is the pair

$$\mathcal{L}(\mathbf{v}) := (\mathcal{V}^*(\mathcal{E}_R(\mathbf{q}, \mathcal{M})), \mathcal{E}_R(\mathbf{q}, \mathcal{M})) \ .$$

3.5.3 Triangulation Update

After the calculation of the projected vertex neighborhood $\mathcal{C}(\mathbf{v})$ and edge neighborhood $\mathcal{L}(\mathbf{v})$, the mesh is refined by adding and removing edges around the examined vertex $\mathbf{v}$. Let $e(\tilde{\mathbf{v}}^*)$ denote a candidate edge with

$$e(\tilde{\mathbf{v}}^*) := \overline{\mathbf{0}\tilde{\mathbf{v}}^*} \quad \tilde{\mathbf{v}}^* \in \mathbb{R}^2$$

with the edge length

$$\|e(\tilde{\mathbf{v}}^*)\| = \|\tilde{\mathbf{v}}^*\| \ .$$

Further, let $\mathcal{E}_\mathcal{C}(\mathbf{v})$ denote the set of candidate edges

$$\mathcal{E}_\mathcal{C}(\mathbf{v}) := \{e(\tilde{\mathbf{v}}^*) \in \mathbb{R}^2 \mid \tilde{\mathbf{v}}^* \in \mathcal{C}(\mathbf{v})\}$$

with $n := |\mathcal{E}_\mathcal{C}(\mathbf{v})|$ and let $\vartheta(\ \mathcal{E}_\mathcal{C}(\mathbf{v})\ )$ denote the set of the elements of $\mathcal{E}_\mathcal{C}(\mathbf{v})$ sorted w.r.t. their edge length

$$\vartheta(\ \mathcal{E}_\mathcal{C}(\mathbf{v})\ ) := \{e_1, \ldots, e_n \in \mathcal{E}_\mathcal{C}(\mathbf{v})\}\ ,\ \forall i = 1, \ldots, n-1 : \|e_i\| \leq \|e_{i+1}\|$$

During the triangulation update, every candidate edge $e \in \vartheta(\ \mathcal{E}_\mathcal{C}(\mathbf{v})\ )$ is tested for integration into the projected set of edges $\mathcal{L}(\mathbf{v})$, using the following rule:

Definition 3.3 *A candidate edge $e \in \vartheta(\mathcal{E}_\mathcal{C}(\mathbf{v}))$ is a valid edge in $\mathcal{L}(\mathbf{v})$, if it has no intersection with any edge $e_\mathcal{L} \in \mathcal{L}(\mathbf{v})$ or if all intersecting edges $e_\mathcal{L} \in \mathcal{L}(\mathbf{v})$ are longer than e.*

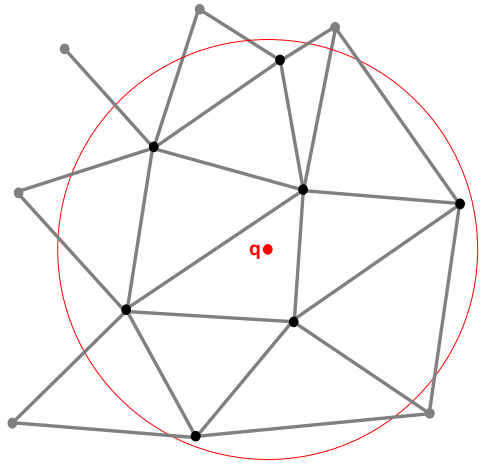
The above rule requires a test for intersection between two edges. Here, a specialized test is used, exploiting the fact that the candidate edge always starts at the origin of the projected coordinate systems. It is described in Appendix A.3. The test result can be divided in three cases:

1. **No intersection**
   The edge e is valid and is added to $\mathcal{L}$.

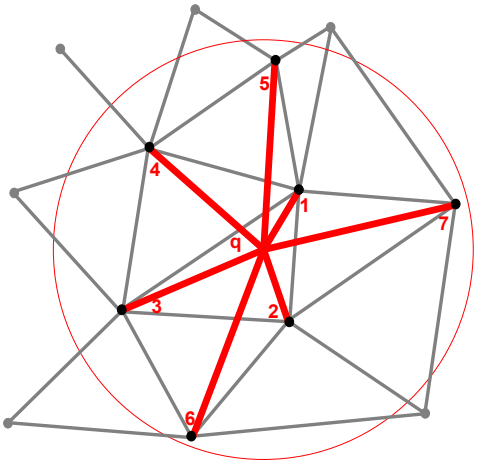
2. **All intersecting edges are longer than** e
   The edge e is valid and is added to $\mathcal{L}$. All intersecting edges are deleted from $\mathcal{L}$.

3. **At least one intersecting edge is shorter than** e
   The edge e is invalid and is not inserted into $\mathcal{L}$.

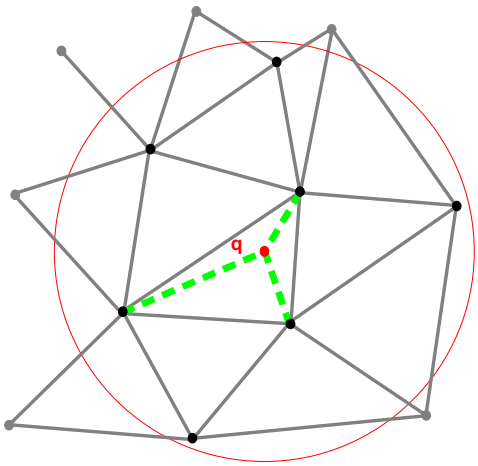
The above test is applied iteratively to all edges in $\vartheta(\mathcal{E}_\mathcal{C}(\mathbf{v})$. Testing the edges in ascending order w.r.t. their length is mandatory, i.e. the test has to start with the shortest edge, because a short edge possibly removes an existing (longer) edge, that otherwise would lead to a rejection of a subsequent longer candidate edge. The entire process of local edge update is summarized in Alg. 3 and is illustrated in Fig. 3.8.



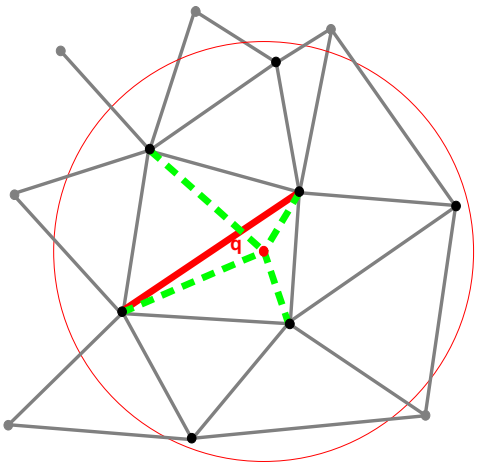
(a) New vertex **q** and projected neighborhood



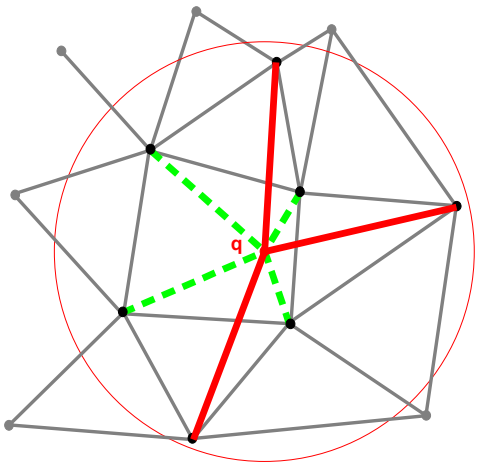
(b) Candidate edges, sorted by length



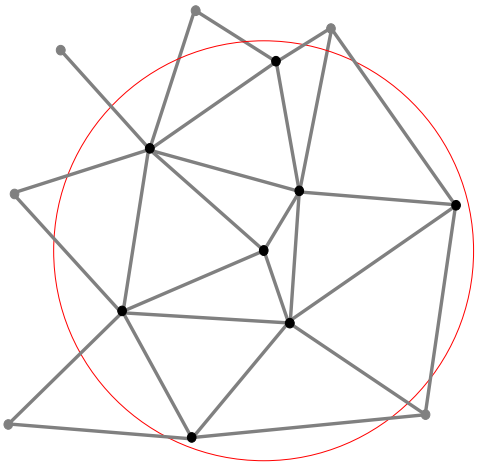
(c) Adding edges 1-3 (No intersection)



(d) Adding edge 4 and removing the intersecting edges (New edge is shorter)



(e) Rejecting edge 5-7 (New edge is longer)



(f) Final locally refined mesh

Figure 3.8: Local update of triangulation by the new vertex **q**. The black points represent the point neighborhood, the gray edges are the edge neighborhood.

Algorithm 3 Incremental update of local edges at the new vertex $\mathbf{v}$ (triangulation update).

```
boolean valid;
boolean intersects;
for all e ∈ ϑ( E_C(v) ) do
  valid := true;
  for all e_L ∈ L(v) do
    intersects := intersect(e,e_L) /* See Alg. 6 in Chapter A.3 */
    if intersects = true then
      if ‖e‖ < ‖e_L‖ then
        Store edge e_L in E_del
      else
        valid = false; break;
      end if
    end if
  end for
  if valid = true then
    Remove all edges in E_del from L(v)
    Add e to L(v)
  end if
  Clear E_del
end for
```

3.5.4 Calculation of Triangle Faces

After the edges of the mesh have been locally updated in the triangulation update step, the modified edge neighborhood $\mathcal{L}(\mathbf{v})$ is transformed back to 3D space and synchronized with the set of edges $\mathcal{E}_\mathcal{M}$. Further, the set of triangles $\mathcal{T}_\mathcal{M}$ has to be updated. Let $\mathcal{E}_{new}$ denote the set of newly inserted edges and let $\mathcal{E}_{del}$ denote the set of removed edges. The removal of an edge implicates the invalidation of the attached triangles and further leads to an incorrect triangle information in the two remaining edges that shared the now invalid triangle. Hence, let $\mathcal{E}_{invalid}(\mathcal{E}_{del})$ be the set of edges with corrupted triangle information. Consequently, the set of modified edges $\mathcal{E}_{mod}$ is defined as

$$\mathcal{E}_{mod} := \mathcal{E}_{new} \cup \mathcal{E}_{invalid}(\mathcal{E}_{del})$$

For every edge $e \in \mathcal{E}_{mod}$, the triangle information has to be updated, i.e. the left and right triangle vertex $\mathbf{v}_l$ and $\mathbf{v}_r$ has to be re-determined, because the triangles are stored via their corresponding edges, as described in Section 3.1. Further, every vertex $\mathbf{v} \in \mathcal{V}_\mathcal{M}$ holds a set $\mathcal{E}(\mathbf{v})$ with all edges that start or end in $\mathbf{v}$.
Let $\mathcal{V}_\mathcal{E}(\mathbf{v})$ denote the set of vertices that are connected to $\mathbf{v}$ by an edge

$$\mathcal{V}_\mathcal{E}(\mathbf{v}) := \{\tilde{\mathbf{v}} \in \mathcal{V}_\mathcal{M} \mid \exists e \in \mathcal{E}(\mathbf{v}) : \tilde{\mathbf{v}}, \mathbf{v} \in e\} \ .$$

The set of candidate triangle points $\mathcal{C}_\mathcal{T}(e)$ of an edge $e \in \mathcal{E}_{mod}$ is the set of all vertices that both edge points are connected to

$$\mathcal{C}_\mathcal{T}(e) := \mathcal{V}_\mathcal{E}(\mathbf{v}_i) \cap \mathcal{V}_\mathcal{E}(\mathbf{v}_j) \quad \text{with } e = \overline{\mathbf{v}_i\mathbf{v}_j} \ .$$

Every triangle $t = \Delta(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)$ with $\mathbf{v}_k \in \mathcal{C}_\mathcal{T}(e_{ij})$ forms either a left or a right triangle w.r.t. e. This is verified by the following rule:

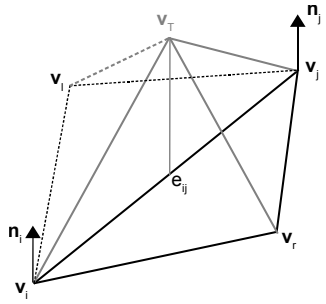


Figure 3.9: The signed tetrahedron volume of the candidate triangle formed by the edge e_{ij} and candidate point $\mathbf{v}_l$ or $\mathbf{v}_r$ respectively. The tetrahedron volume $V_{\text{Tet}}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_r, \mathbf{v}_T)$ is negative and the volume $V_{\text{Tet}}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_r, \mathbf{v}_T)$ is positive.

Definition 3.4 *A point $\mathbf{v}_k$ defines a left triangle with respect to the edge $e = \overline{\mathbf{v}_i\mathbf{v}_j}$, if the signed tetrahedron volume $V_{Tet}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_T)$ with the tetrahedron point $\mathbf{v}_T = \frac{\mathbf{v}_i+\mathbf{v}_j}{2} + \mathbf{n}_i + \mathbf{n}_j$ is positive.*

The signed volume V_{Tet} is given by

$$V_{\text{Tet}}(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) = \frac{1}{6} \det \begin{pmatrix} (\mathbf{v}_1 - \mathbf{v}_0)^T \\ (\mathbf{v}_2 - \mathbf{v}_0)^T \\ (\mathbf{v}_3 - \mathbf{v}_0)^T \end{pmatrix} ,$$

as defined in the textbook of O'Rourke [O'R98]. The Definition 3.4 is further visualized in Fig. 3.9.
If more than one candidate point for the left or right triangle exists in $\mathcal{C}_T(e)$, the triangle with the smallest volume V_{Tet} is used. Consequently, the left triangle point is given by

$$\mathbf{v}_l(e) := \underset{\mathbf{v}_k \in \mathcal{C}_T(e)}{\arg\min} \{V_{\text{Tet}}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_T) > 0\}$$

and the right triangle point is

$$\mathbf{v}_r(e) := \underset{\mathbf{v}_k \in \mathcal{C}_T(e)}{\arg\min} \{V_{\text{Tet}}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_T) < 0\} \ .$$

3.6 Summary and Discussion

In this Chapter, the real time streaming surface reconstruction *RT-SSR* method for visual feedback tasks is introduced. A triangle mesh is generated by successively adding new points to a processing pipeline.
The processing pipeline consists of two principal stages: the normal estimation stage and the mesh generation stage. In the first stage the surface normals for every inserted point are estimated by least-square fitting a tangent plane through a local point neighborhood of the examined point. Here, a combination of ball neighborhood and k-nearest neighbors is applied. With every insertion of new points, the neighborhood is adapted for every point individually. The normal estimation for each point is controlled by an adaptive neighborhood and a global limitation of the point density. The estimation results are validated and changes are tracked, thus, only plausible points are transfered to the subsequent mesh generation stage. During the mesh generation stage, the new points are inserted as vertices of the emerging mesh. For every newly inserted vertex a localized 2D triangulation is performed on the projection of a local ball neighborhood.

The core of the reconstruction process is the estimation of surface normals, because incorrect normals result in a faulty projection in the mesh generation stage and thus lead to a corrupted triangulation. The first step of this estimation is a global density limitation. The problem of an unwanted dependency on the input order, which potentially causes the rejection of points with high quality by ones with a lower quality, is overcome by introducing a replacement limitation. However, this replacement limitation requires an additional coordinate per sample point an thus increases the memory consumption and computational effort. Therefore, the limitation method should be selected according to the characteristics of the measured data. A replacement limitation is only applied, if a suitable quality criterion exists and varies significantly.
The limitation is parametrized by a minimum distance $R_{\min}$. This radius has to be chosen carefully, if local surface details are not to be omitted. The measurable detail of a surface depends on the sample density and the accuracy of the scanner system used. Hence, the minimal expected deviation $\tilde{\sigma}_{\min}$ can be used to adjust the radius.
The normal estimation is controlled by the maximum number of neighboring points and the initial radius. This k-in-R neighborhood is adapted for each point and with every update step. This concept allows for an initial coarse estimation of a surface normal, even for locally sparse sampled point sets. If the sample density rises, the radius of its neighborhood shrinks, resulting in normals that represent the local shape of the surface more precisely. This approach couples the parameters of the neighborhood k and R_{n0} to the limitation radius $R_{\min}$. Hence, the combination of parameters has to be chosen carefully. This aspect is further discussed in Chapter 5.
During the selection step, the estimated surface normals are validated, ensuring that only plausible surface normals are used in the subsequent mesh generation. The selection implicitly filters outliers and very sparse areas from the point set, if this neighborhood is too small or the variances are bad, resulting in non-plausible estimates. This filtering is rather strict and can also delay or prevent the further processing of points with good estimates. In this work, a fast selection bypass is used to accelerate the selection of new points that are already embedded in a neighborhood of points with valid surface normals. Again, the fast selection has to be configured carefully, as otherwise points with invalid normals are transfered to the mesh generation. This can be eased by tracking the surface normals, which does, however, increase computational costs.
The mesh generation stage has an additional limitation step that allows for controlling the resolution of the generated mesh without influencing the process of normal estimation. For triangulation, the edge- and point neighborhood of the resulting mesh is used. Similar to the normal estimation stage, the limitation radius and the neighborhood radius are coupled.
The triangulation update is an expensive operation. The correctness of this refinement depends on the estimated surface normal. Re-insertion of a vertex necessitates to the deletion of the old vertex and its connectivity, to re-insert the new vertex, and to update the mesh. Hence, the selection step in the normal estimation stage should be strict enough to avoid unnecessary re-insertions of vertices, which interferes with the requirement of fast mesh generation and the filtering problem stated above.
The complexity of both stages is limited, because all operations are performed on local neighborhoods with an upper bound for the number of points. The only operations that have to be performed on the complete data set are the insertion of points and the query of a ball neighborhood. The acceleration of these global operations are the topic of the following Chapter 4.

4
Spatial Data Structures

The streaming surface reconstruction method mostly performs operations on local subsets of the stored data sets, however, at least the determination of these subsets is a global operation. Hence, a data structure that accelerates the operations performed on the used point sets and meshes is required. During manual scanning and streaming processing of real time measurements, an appropriate data structure needs to reflect the fact that the data is incrementally growing with every measurement. Moreover, no a priori knowledge concerning the object's size or number of sample points can be assumed. Thus, a data structure must be able to dynamically extend itself and the space it covers.
The acceleration of global operations on the stored data of the streaming surface reconstruction is the topic of this Chapter. First, general spatial data structures for a dynamic purpose are discussed and two suitable structures are introduced. Further, their application to the streaming surface reconstruction process is explained. The advantages and disadvantages of the two data structures are analyzed and the best suitable structure w.r.t. to the size of the scanned scene or object is recommended.

4.1 Dynamic Space Partitioning

Spatial data structures are used in numerous applications, especially in the fields of computer vision and computer graphics. They are commonly used either to reduce the complexity of irregular 3D data by approximation with regular structures or to provide a fast access to local features by exploiting the spatial ordering of the structure. In this work, the spatial data structure enables for fast operations on point sets and meshes that are dynamically changing. In the following section, related work is discussed and two suitable spatial structures, the **Dynamic Voxel Space** and the **Extendable Octree**, are presented.

4.1.1 Related Work

The general concept of every spatial data structure is to partition the space into finite subsets and to provide a rapid access to these elements. Existing approaches can be

categorized into structures with homogeneous partitioning and such using heterogeneous or irregular partitioning. Examples for homogeneous partitioning in 3D are the *linear voxel space* or the *octree*. Structures with irregular partitioning are e.g. the *BSP-trees*[1] or the *R-tree* [Gut84] and its variants.

The R-tree has been first published in the work of Guttman [Gut84]. This tree and its variants, e.g. the R^*-tree or the R^+-tree, are often used for GIS[2]. In the early textbook of Samet [Sam90], different types of quadtrees and octrees are intensively described and discussed in the context of image processing and GIS systems. Here, the spatial structures are primarily designed for compression and fast access to static data sets. In the textbook of Akenine-Moeller [AMH02], spatial data structures, e.g. BSP trees and octrees, are used for accelerated rendering techniques, e.g. culling algorithms, intersection tests, ray tracing and collision detection. Rusinkiewicz [Rus01] uses a BSP-tree for designing of a point rendering technique, called QSplat. Here, a point set is divided into a hierarchy of bounding spheres. Instead of rendering the points, the tree is traversed top down, until a sphere not longer contributes to the rendering process, because it is not in the viewing frustum or it is to small for the given view. The technique is extended to a network streaming variant, i.e. subtrees are loaded via a network stream if required.

Spatial data structures are also used in the context of surface reconstruction. In the work of Hoppe [Hop94] a single-resolution voxel space is used for approximating an unorganized point set. The voxels are labeled by an implicit function that represents the sampled surface. Finally, an iso-surface is extracted from the voxel space by using the marching cubes algorithm of Lorensen and Cline [LC87].

Generally, irregular partitioning provides a better partitioning of the data than a homogeneous partitioning. However, the computational effort for re-balancing the structure during modification of the stored data (insertion or removal of elements) is higher for structures with irregular partitioning. Hence, the structures applied in this work are based on homogeneous partitioning, as the data is dynamically changing and the overall computation time should not be increased by unnecessary restructuring of the data storage.

In detail, the $\mathbb{R}^3$ space is partitioned into an axis-aligned, regular grid of cubes and a fast access to these discrete elements is established. The homogeneous partitioning is called **voxelization**, the discrete volume elements are named **voxels**[3]. Further, voxel spaces can be divided into single-resolution and multi-resolution spaces. In the first case, all voxels have the same edge length. In the second case, the space is hierarchically constructed and multiples of the basic edge length are possible.

Every voxel stores information related to the covered volume. In many applications, a voxel stores a binary value that indicates whether the volume is inside or outside of an object (e.g. in the work of Hoppe [Hop94]). In this work, a voxel is used as a container that stores the points, vertices and edges of the streaming surface reconstruction process. In the following, two approaches for dynamically creating the voxel space and rapidly accessing the voxels are introduced: the single-resolution **Dynamic Voxel Space** and the multi-resolution **Extendable Octree**.

[1] BSP-Tree = Binary Space Partitioning Tree (The best known BSP-tree is the kd-tree, see e.g. the textbook of Samet [Sam90])

[2] GIS = Geographic Information System

[3] The word *voxel* is a portmanteau of the words volumetric and pixel (From `http://en.wikipedia.org/wiki/Voxel`, 2008).

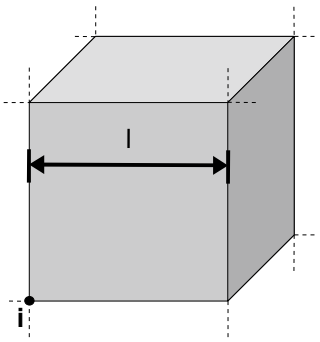


Figure 4.1: Definition of a voxel: A voxel is a cubical volume defined by an edge length or voxel size l and the grid coordinate at the lower, left, and front corner.

4.1.2 Dynamic Voxel Space

Various approaches to design a single-resolution voxel space exist. A simple and fast implementation is the mapping to a one-dimensional, connected field (in memory), called **linear voxel space**. Each voxel is mapped to a unique and scalar index, e.g. by a x-y-z-order encoding[4], Z-ordering or Hilbert-ordering. A detailed description of linear voxel spaces is given in the textbook of Samet [Sam90]. The advantage of this voxel space implementation is the random and fast access to all elements. As a drawback, the total volume covered has to be known a priori. Further, all voxel of the volume must be allocated in memory. This non-dynamic and memory consuming design is not suitable for growing data sets, as required in this work.

The **Dynamic Voxel Space** is a dynamic and sparse single-resolution voxel space i.e. the volume does not need to be known a priori and only *non-empty* voxels have to be stored in memory. In detail, the voxels are stored in a self-balancing binary search tree[5]. This approach has already been presented in the context of surface reconstruction by Bodenmüller [BH04].
Let l denote the edge length or size of an voxel in the $\mathbb{R}^3$ space and let $\mathbf{i} \in \mathbb{Z}^3$ denote the **grid coordinate** of a voxel on the discrete grid. The grid coordinate is the lower, left, and front corner of the cubical voxel volume, as illustrated in Fig. 4.1. Further, let $\mathbf{c}_0$ denote the origin of the voxel space in $\mathbb{R}^3$ i.e. the location of the grid coordinate $\mathbf{i} = (0\ 0\ 0)^T$.[6] Hence, a voxel can be described in the $\mathbb{Z}^3$ by its grid coordinate. The corresponding voxel or grid coordinate $\mathbf{i} \in \mathbb{Z}^3$ of a point $\mathbf{p} \in \mathbb{R}^3$ is given by

$$\mathbf{i} : \mathbb{R}^3 \to \mathbb{Z}^3 \,;\, \mathbf{p} \mapsto \mathbf{i}(\mathbf{p}) := \left\lfloor \frac{\mathbf{p} - \mathbf{c}_0}{l} \right\rfloor = \begin{pmatrix} \lfloor (p_x - c_{0x})/l \rfloor \\ \lfloor (p_y - c_{0y})/l \rfloor \\ \lfloor (p_z - c_{0z})/l \rfloor \end{pmatrix},$$

with the *floor-function*

$$\lfloor x \rfloor = \max \{ n \in \mathbb{Z} \mid n \leq x \} \ .$$

Correspondingly, the coordinate $\mathbf{c} \in \mathbb{R}^3$ of a voxel with grid coordinate $\mathbf{i} \in \mathbb{Z}^3$ is

$$\mathbf{c} : \mathbb{Z}^3 \to \mathbb{R}^3 \,;\, \mathbf{i} \mapsto \mathbf{c}(\mathbf{i}) = \mathbf{i} \cdot l + \mathbf{c}_0 \ .$$

[4] The x-y-z-order is also known as lexicographic ordering (see Definition 4.1).

[5] In this work a *red-black tree* implementation of a 2-3-4-Top-Down tree is used. Other concepts like the AA tree or AVL tree can be used alternatively. More information can be found e.g. in the textbook of Sedgewick [Sed92].

[6] The shift $\mathbf{c}_0$ of the voxel space origin is optional and can be used to keep the absolute values of $\mathbf{i}$ low and thus to avoid numerical problems.

Insertion The tree is constructed dynamically, starting with a void volume. It grows with every new voxel that is inserted. Each voxel represents a node in the balanced search tree. For the insertion of new voxel into the tree and also for queries, a unique comparison operation between voxels is required. In this work, the **lexicographical ordering** is used for comparison and defined by the following rule:

Definition 4.1 *Let* $\mathbf{i}_1 = (u_1, v_1, w_1)^T, \mathbf{i}_2 = (u_2, v_2, w_2)^T \in \mathbb{Z}^3$ *denote the grid coordinates of two voxels. The voxel* $\mathbf{i}_1$ *is lexicographically smaller than* $\mathbf{i}_2$ *(*$\mathbf{i}_i < \mathbf{i}_2$*) if and only if*

$$u_1 < u_2 \vee (u_1 = u_2 \wedge (v_1 < v_2 \vee (v_1 = v_2 \wedge w_1 < w_2))) .$$

Query The query of a single voxel or node in a binary search tree is performed by traversing the tree top-down until the desired element is found. Here, the comparison operation as defined in 4.1 is used for selecting the correct branch during tree traversal.
In the context of the *RT-SSR* method, the query of all voxel inside an *Axis-Aligned Bounding Box (AABB)* is required, as described in the next Section 4.2. Let $\mathcal{D}$ denote the set of all non-empty (or allocated) voxel in the space

$$\mathcal{D} := \{\mathbf{i}_1, \ldots, \mathbf{i}_n\} \in \mathbb{Z}^3$$

and let $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$ denote the minimum and maximum coordinates of the *AABB*. Further, let $\mathbf{i}_{\min} = \mathbf{i}(\mathbf{p}_{\min})$ and $\mathbf{i}_{\max} = \mathbf{i}(\mathbf{p}_{\max})$ the corresponding grid coordinates. Hence, the set of voxel that intersects with the *AABB* is given by

$$\mathcal{D}_{AABB} = \{\mathbf{i} \in \mathcal{D} \mid \forall x = 1, 2, 3 \,:\, \mathbf{i}_{\min_x} \leq \mathbf{i}_x \leq \mathbf{i}_{\max_x}\} \ . \tag{4.1}$$

The query of the set $\mathcal{D}_{AABB}$ requires a separate search of each voxel in the set. However, in Section 4.2 an extension to the Dynamic Voxel Space that allows for the query of a set of direct neighboring voxels with only a single tree traversal is discussed.

The self-balancing strategy of the tree guarantees that an insert- or query- operation has a complexity of $\mathrm{O}(\log_2 N)$ with N being the number of elements in the tree. More details on operations on self-balancing trees can be found e.g. in the textbook of Sedgewick [Sed92].
The advantage of this implementation is that only non-empty voxels are allocated in memory. Thus, the volume does not have to be known a priori and only a small memory overhead for the tree is required. A drawback of this structure is the loss of spatial order, because the voxels are mapped to a one-dimensional search tree. Hence, spatially neighboring voxels are usually not mapped to neighboring nodes in the binary tree, as illustrated in Fig. 4.2.

4.1.3 Extendable Octree

An octree is a hierarchical data structuring technique that bases on uniform, recursive decomposition. It is commonly used to compress information by merging voxels with equal feature values into one larger voxels. Various applications of octrees with different designs exist in computer vision and computer graphics. Examples for octree applications can be found in the textbooks of Samet [Sam90] and Akenine-Moeller [AMH02]. Typically, a cubic root volume is subdivided into eight smaller cubes. This is repeated until a specified resolution is reached. This concept is illustrated in Fig. 4.3.
An octree can either be implemented as linear octree that stores all nodes in a linear

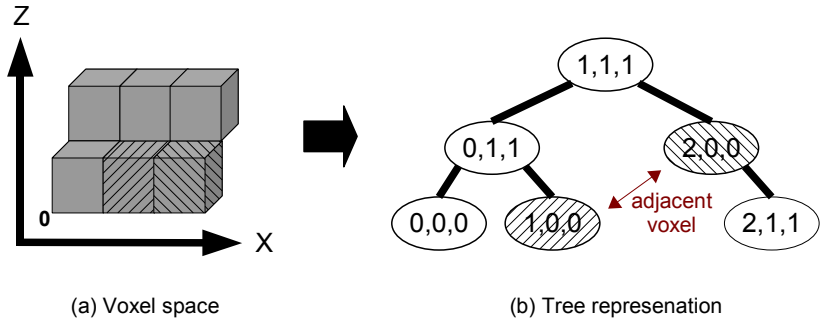


Figure 4.2: Binary search tree representation of a voxel space: A set of voxels (a) and their representation in the binary search tree. Neighboring voxels typically are not mapped to neighboring nodes in the tree.

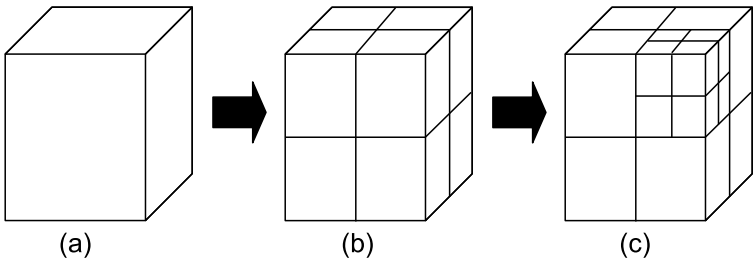


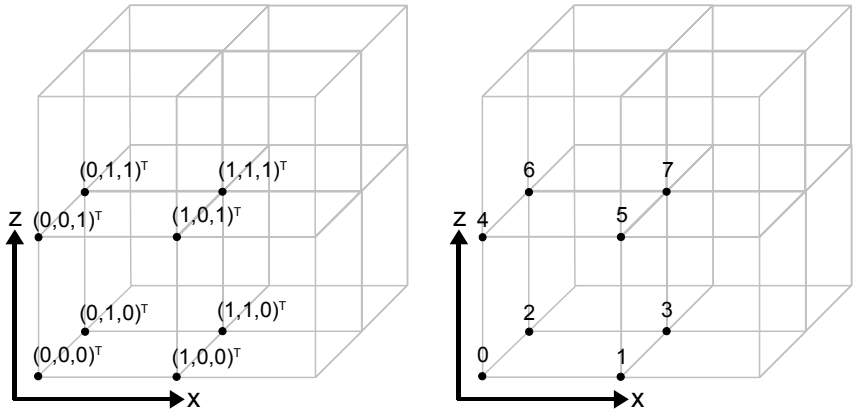
Figure 4.3: Basic concept of an octree: (a) represents a 'root' volume, in (b) the 'root' volume is subdivided into smaller values, in (c) one subdivided volume is subdivided further.

array, or as direct octree that allocates the tree nodes separately and links the memory addresses. The linear implementation implicates that the covered volume has to be known a priori, analogous to the linear voxel space. Contrary, the direct octree can be modified dynamically while the access is slower w.r.t. the linear design, since a voxel access always requires a complete tree traversal.
The **Extendable Octree** is a direct octree but uses an inverse creation strategy compared to usual strategies, i.e. the octree initially has a void volume and is constructed bottom-up incrementally. It grows with every insertion of a new non-empty voxel, similar to the creation strategy of the Dynamic Voxel Space.
The uppermost voxel of the octree is denoted as **root**, the lowest voxels are denoted as **leaves**. Each voxel has a level k that represents its position in the hierarchy, additionally to its grid location $\mathbf{i}$. The leaf voxels are at level $k = 0$, the root voxel is at level k_r. In comparison to a single-resolution voxel space, the octree allows voxels with different edge lengths. The smallest edge length l_0 of the leave voxels characterizes the octree similar to the length l in the single-resolution voxel space. The edge length l_k of a voxel with level k is

$$l_k = l_0\, 2^k \quad .$$

Every non-leaf voxel has eight connections (pointers) to its **children** or **subvoxel**. The location of a subvoxel can be described in relation to the coordinate system of its parent voxel by the vector $\Delta\mathbf{i}_c$, as illustrated in Fig. 4.4(a). Moreover, this subvoxel coordinate $\Delta\mathbf{i}_c$ can be mapped to a scalar index i_c that allows for uniquely indexing the subvoxel w.r.t. its parent. This subvoxel encoding is further visualized in Fig. 4.4(b). The encoding is related to the relative location vector by

$$i_c(\Delta\mathbf{i}_c) := \left\langle (2^0\ 2^1\ 2^2)^T, \Delta\mathbf{i}_c \right\rangle \quad . \tag{4.2}$$



(a) child location vector $\Delta\mathbf{i}_c$ (b) spatial index i_c

Figure 4.4: Access to the children of an octree voxel: The location $\Delta\mathbf{i}_c$ of a child voxel relative to its parent voxel (a) and its unique labeling by the corresponding spatial index i_c (b).

Query A query of a voxel $\mathbf{i}$ at level k is performed by traversing the octree top-down. Therefore, the scalar index of the next subvoxel to descend to has to be determined in every step. Let $\tilde{\mathbf{i}}$ denote the current voxel of the traversal at level $\tilde{k} > 0$ and let $\mathbf{i}$ be a direct or indirect subvoxel with $k < \tilde{k}$. Then, the scalar index i of the next subvoxel in the direction of the desired voxel $\mathbf{i}$ is

$$i = \text{child}(\tilde{\mathbf{i}}, \tilde{k}, \mathbf{i}) := i_c \left(\left\lfloor \frac{2(\mathbf{i} - \tilde{\mathbf{i}})}{2^{\tilde{k}}} \right\rfloor \right) \quad . \tag{4.3}$$

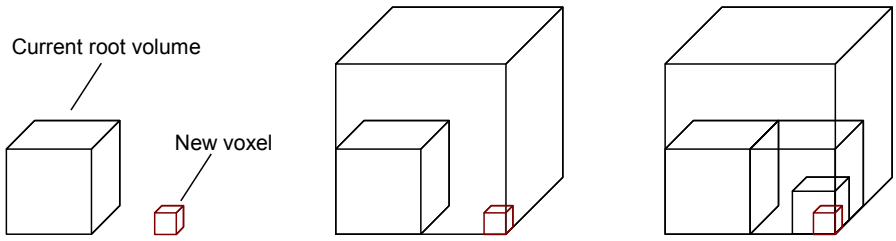
Let $\mathcal{D}_k$ denote the set of all non-empty voxels with level k. Hence, the set $\mathcal{D}_0$ is comparable to the set $\mathcal{D}$ of the *Dynamic Voxel Space*. Here, $\mathcal{D}$ denotes the set of all non-empty voxels at all levels,

$$\mathcal{D} := \bigcup_{k=0}^{k_r} \mathcal{D}_k \quad .$$

In comparison to a single-resolution voxel space, the octree supports the combined query of all voxel in an *AABB*. The set of voxels $\mathcal{D}_{AABB} \subseteq \mathcal{D}$ intersecting with the *AABB* is collected by traversing the octree top-down, analogous to a single-voxel query. However, at each traversal step, the intersection between the current voxel and the *AABB* defined by $\mathbf{i}_{\min}$ and $\mathbf{i}_{\max}$ must be determined. If an intersection exists, the branch is further traversed. Let $\tilde{\mathbf{i}}$ denote the current voxel of the traversal at level $\tilde{k}$. The intersection between the current voxel and the search volume can be tested by the following rule:

Definition 4.2 *An intersection between a voxel $\tilde{\mathbf{i}}$ at level $\tilde{k}$ and a grid volume from $\mathbf{i}_{\min}$ to $\mathbf{i}_{\max}$ exists, if one of the points $\mathbf{i}_{\min}$ and $\mathbf{i}_{\max}$ is inside the AABB defined by $\tilde{\mathbf{i}}$ and $(\tilde{\mathbf{i}} + \mathbf{2}^{\tilde{k}})$, or if one of the points $\tilde{\mathbf{i}}$ and $(\tilde{\mathbf{i}} + \mathbf{2}^{\tilde{k}})$ is inside the AABB defined by $\mathbf{i}_{\min}$ and $\mathbf{i}_{\max}$.*

Insertion The insertion of a new leaf voxel consists of two steps. First, the tree has to be **extended**, i.e. new root voxels are added to the existing root, until the root volume covers the new voxel. Then, the new root is **subdivided** until the root node is connected to the new voxel via its children. This dynamic construction is illustrated in Fig. 4.5. In the following, the two basic operations **extension** and **subdivision** are explained in detail.



(a) new voxel (b) extension (c) subdivision

Figure 4.5: Construction of the Extendable Octree: A new voxel (red) is inserted (a). First, the octree is extended until the new voxel is inside the root volume (b), i.e. the current root voxel is embedded into a new root volume. Then, the octree is subdivided until the new root voxel is connected to the newly inserted voxel (c).

Extension Let $\mathbf{i}$ denote the grid coordinate of a new leaf voxel that is not inside the root volume of the octree. The extension operation incrementally extends the root volume of the octree by embedding the original octree into a new root volume (of double edge length). This operation is detailed by the pseudo-code in Alg. 4.

Algorithm 4 Octree extension for the insertion of a new leaf voxel with grid coordinate $\mathbf{i}$

```
NodePtr root := Octree.root
while not inside(newVoxel , root) do
  NodePtr newRoot := create_node() /* create new node */
  newRoot.k := root.k + 1 /* increase level */
  newRoot.i := loc(root.i, root.k, i) /* see Equation (4.4) */
  i_c := child(newRoot.i, newRoot.k, root.i) ) /* see Equation (4.3)) */
  newRoot.child[i_c] := root /* link old and new root */
  root := newRoot
end while
Octree.root := root
```

The *inside*-condition in the while-loop is a test for intersection between two axis-aligned bounding boxes[7]. Hence, a simple comparison rule can be used for testing:

Definition 4.3 *A leaf voxel* $\mathbf{i}$ *is inside the volume of a root voxel* $\mathbf{i}_r$ *and level* k_r*, if*

$$\max(\mathbf{i}_r - \mathbf{i}) < 2^{k_r} \wedge \min(\mathbf{i}_r - \mathbf{i}) \geq 0 \ .$$

At each iteration, the grid coordinate of the new root voxel $\mathbf{i}_{new}$ must be determined. Here, the voxel is grown into the direction of the new leaf voxel by

$$\mathbf{i}_{new} = \mathrm{loc}(\mathbf{i}_r, k_r, \mathbf{i}) := \mathbf{i}_r - (\mathbf{1} - \Theta(\mathbf{i} - \mathbf{i}_r) \cdot 2^{k_r}) \tag{4.4}$$

with the element-wise *unit step function*

$$\Theta(x) := \begin{cases} 1 & x \geq 0 \\ 0 & \text{else} \end{cases} \qquad x \in \mathbb{R} \ .$$

[7] See the textbook of Bergen [Ber03] for further information on intersection tests.

Subdivision Let $\mathbf{i}$ denote the grid coordinate of a voxel with level k that is inside the root volume but not integrated into the octree structure. The **subdivison** operation descends through the octree and creates top-down all non-existing nodes until the voxel is connected to the octree's root voxel. The subdivision operation is detailed by the pseudo-code in Alg. 5.

Algorithm 5 Octree subdivision for a new voxel with grid coordinate $\mathbf{i}$ and level k

```
Require: New voxel is inside the root volume
  VoxelPtr voxel := Octree.root
  while voxel.k > k do
    [i_c, Δi_c] := child(voxel.i, voxel.k, i) /* see Equation (4.3) */
    if not voxel.child[i_c] then
      voxel.child[i_c] := create_node() /* Create child voxel */
      voxel.child[i_c].i := node.i + Δi_c
      voxel.child[i_c].k := voxel.k - 1
    end if
    voxel := voxel.child[i_c] /* descend one level */
  end while
  return voxel
```

4.2 Application to Streaming Modeling

The streaming modeling method introduced in this work consists of two principle stages, the **normal estimation stage** and the **mesh generation stage**, as described in Chapter 3. In both stages, a fast access to dynamically changing spatial data sets is required. New points are inserted and local neighborhood queries must be performed in the normal estimation stage. During mesh generation, vertices and edges are inserted, their neighborhoods are searched, and edges may even be deleted. In the following, the acceleration of these operations by applying the previously introduced spatial data structures is discussed.

4.2.1 Operations on Points and Vertices

Points and vertices have no spatial extension and thus are stored in the corresponding (leaf) voxel. Therefore, each voxel $\mathbf{i}$ holds a list[8] $\mathcal{P}_\mathbf{i}$ containing the points or vertices that are inside it, with

$$\mathcal{P}_\mathbf{i} := \{\mathbf{p} \in \mathcal{P} \mid \mathbf{i}(\mathbf{p}) = \mathbf{i}\} \ .$$

In the following, insert- and query-operations are explained using a sample point set $\mathcal{P}$ as an example. The operations on a set of vertices $\mathcal{V}$ are performed similar.

Point Insertion The insertion of a point or vertex into a data set is performed by searching for or creating the corresponding voxel and inserting the point into the voxel's list. The complexity of insertion depends only on the complexity of searching for or creating the corresponding voxel, as the insertion into the list has a constant complexity O(const.).

[8] A list is a simple data structure that allows for fast insertion and deletion as well as linear access. Detailed information can be found e.g. in [Sed92]

Neighborhood Query The query of the ball neighborhood $\mathcal{N}_R(\mathbf{q}, \mathcal{P})$ for a point $\mathbf{q}$ is the search of all points in a spherical volume around $\mathbf{q}$. This query is accelerated by identifying all non-empty voxels that intersect with the spherical volume first, and then testing the points inside these voxels only.

Let $\mathcal{D}_\mathcal{P}$ denote the set of non-empty voxels

$$\mathcal{D}_\mathcal{P} := \{\mathbf{i} \in \mathbb{Z}^3 \mid \mathcal{P}_\mathbf{i} \neq \{\}\} \ .$$

For the *Dynamic Voxel Space*, this set is equal to the set $\mathcal{D}$ from the previous section ($\mathcal{D}_\mathcal{P} = \mathcal{D}$). If the *Extendable Octree* is used, the set is equal to the set of all leaf voxels ($\mathcal{D}_\mathcal{P} = \mathcal{D}_0$), as only leaf voxels can contain points.
For the query of a ball neighborhood, all voxels that intersect with the spherical volume centered at $\mathbf{q}$ and with radius R are required. Here, the *AABB* of the spherical volume is used to find these voxels. Let $\mathcal{D}_\mathcal{N}$ denote the set of all non-empty voxels that intersect with the *AABB* that ranges from $(\mathbf{q} - R)$ to $(\mathbf{q} + R)$. Analogues to Equation (4.1), the set is given by

$$\mathcal{D}_\mathcal{N} = \{\mathbf{i} \in \mathcal{D}_\mathcal{P} \mid \forall x = 1, 2, 3 \ : \ \mathbf{i}_x(\mathbf{q} - R) \leq \mathbf{i}_x \leq \mathbf{i}_x(\mathbf{q} + R)\} \ . \tag{4.5}$$

Hence, the set of points $\mathcal{P}_{\mathcal{D}_\mathcal{N}}$ that must be tested during a neighborhood query is

$$\mathcal{P}_{\mathcal{D}_\mathcal{N}} = \bigcup_{\mathbf{i} \in \mathcal{D}_\mathcal{N}} \mathcal{P}_\mathbf{i} \ .$$

Consequently, the point ball neighborhood $\mathcal{N}_R(\mathbf{q}, \mathcal{P})$ is given by

$$\mathcal{N}_R(\mathbf{q}, \mathcal{P}) \equiv \mathcal{N}_R(\mathbf{q}, \mathcal{P}_{\mathcal{D}_\mathcal{N}}) \ .$$

The complexity of the neighborhood query comprises the complexity of finding $\mathcal{D}_\mathcal{N}$ and the linear complexity $\mathrm{O}(|\mathcal{P}_{\mathcal{D}_\mathcal{N}}|)$ of testing all points in the corresponding point set $\mathcal{P}_{\mathcal{D}_\mathcal{N}}$. The latter has an upper bound, caused by the density limitation pattern introduced in Section 3.2.
Concluding, the complexity of both operations depends on a fast access to the voxel space. In the following, this access and its complexity is discussed for the previously introduced two data structures.

Dynamic Voxel Space

The complexity of insertion and query of a single voxel in a binary search tree is $\mathrm{O}(\log_2 |\mathcal{D}|)$, i.e. the access speed is directly coupled to the number of non-empty voxels in the space. For a neighborhood query, every voxel in the interval $\mathcal{D}_\mathcal{N}$ has to be searched separately, resulting in a complexity of

$$\mathrm{O}(|\mathcal{D}_\mathcal{N}| \log_2 |\mathcal{D}|) \ .$$

In the normal estimation stage, potentially more queries than insertions are performed. Moreover, not every insertion requires the creation of a new voxel. Hence, the overall processing for both stages can be accelerated by storing additional links to the direct neighborhood for every voxel, as illustrated for two dimensions in Fig. 4.6. The spatial order information is partially restored and can be used for the query.
Using this extension, only the central voxel that contains the query point $\mathbf{q}$ has to be searched in the tree, as the direct neighboring voxel can be accessed via this

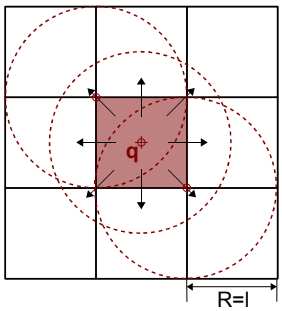


Figure 4.6: Neighborhood query with linked neighboring voxels: In two dimensions, the links to the eight neighboring voxels are stored for each voxel (black arrows). In three dimensions, each voxel has 26 neighbors. Hence, for the query of a ball neighborhood with radius $R \leq l$ only the corresponding voxels (red quadratic area) of the query point $\mathbf{q}$ has to be found. Three neighborhoods with radius $R = l$ are displayed (dashed red circles) as example.

voxel. However, this acceleration can only be applied, if the neighborhood sphere fits completely into the volume of the voxel and its direct neighbors. This can be guaranteed for a ball neighborhood with radius R, if the edge length of the voxels is sufficiently large, i.e. if

$$R \leq l \ .$$

This requirement is visualized in Fig. 4.6. Here, the marginal case of a neighborhood sphere that is centered in a corner of the central voxel is shown. This extension reduces the complexity of a query to

$$\mathrm{O}(\log_2 |\mathcal{D}|) \ .$$

In exchange, the creation of a new voxel implicates an additional search of the 26 neighboring voxels and the storage of the information as link in each voxel. Hence, the complexity of a voxel creation increases to

$$\mathrm{O}(27 \log_2 |\mathcal{D}|) \ .$$

Extendable Octree

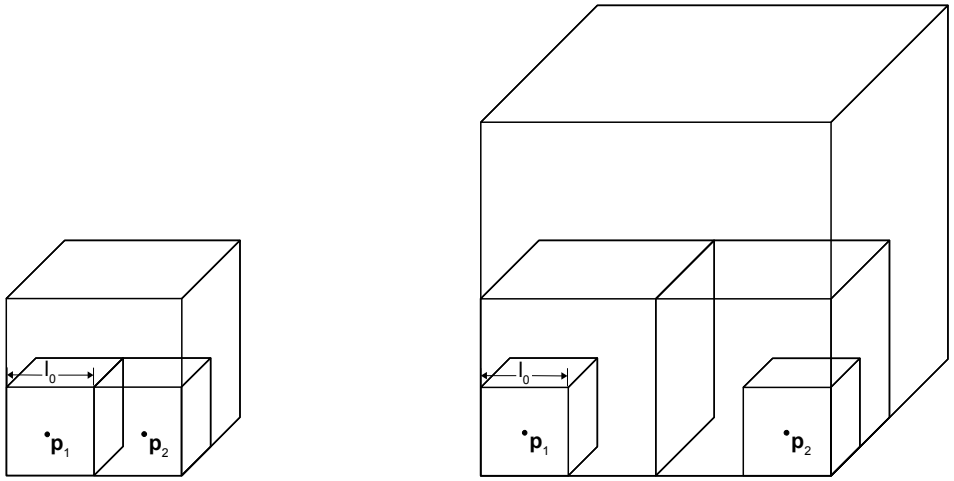
The octree typically reflects the spatial expansion of the stored data, whereas the Dynamic Voxel Space omits this global property. Hence, the number of levels in an octree at a given basic voxel size l_0 is directly coupled to this expansion. The number of steps at the tree traversal for adding new voxels and searching existing voxels or aligned volumes depends on the compactness of the stored data set. This fact is illustrated in Fig. 4.7 for two points with two different distances at a basic voxel size l_0. Thus, the minimum number of levels $k_{\min}$ at a basic voxel size l_0 that are required to cover a data set with an axis-aligned rectangular bounding volume of edge length $l_x \times l_y \times l_z$ is

$$k_{\min} = \lceil \log_2 \max \{l_x, l_y, l_z\} - \log_2 l_0 \rceil \ , \tag{4.6}$$

with the ceiling function

$$\lceil x \rceil = \min \{n \in \mathbb{Z} \mid n \geq x\} \ .$$

However, the insertion of a new point either results in the query of an existing voxel or in the creation of a new one. The complexity of searching an existing leaf voxel is $\mathrm{O}(k_r)$, being k_r the level of the root voxel and thus number of levels in the octree. The creation of a new leaf voxel consists of $(k_r^* - k_r)$ extension steps and k_r^* subdivision



(a) $k = 2$ (b) $k = 3$

Figure 4.7: Complexity of octree traversal: In this example the tree consists of two non-empty voxels (leaf nodes). The number of octree levels k required to integrate both voxels depends on the distribution of the voxels (compactness). In (a) the voxels are directly neighboring, resulting in a tree with $k = 2$ levels. In (b) the voxels are located further apart, resulting in $k = 3$ levels.

steps, with the initial root level k_r and the new root level k_r^*. The level k_r^* can be calculated by Equation (4.6) using the combined bounding box of the original octree and the new point.
Contrary to the Dynamic Voxel Space, the octree enables the search of the set $\mathcal{D}_\mathcal{N}$ at once, as described in Section 4.1. The effort for the tree traversal at a combined query depends on the constellation of the searched voxel in the octree and thus is very variable. However, a separate search of all voxels in $\mathcal{D}_\mathcal{N}$ has a complexity of $\mathrm{O}(k_r|\mathcal{D}_\mathcal{N}|)$ and thus is an upper bound for the effort of the combined query of a set of voxels.

4.2.2 Operations on Edges

The insertion and query of edges is performed synchronously to the point operations. First, the corresponding voxels or voxel set has to be found, then the local operation is performed. However, unlike vertices, edges have an extension in space. Hence, the storage of edges and resulting possible query operations are more complicated.
An edge can be stored explicitly as element in the voxel that fully contains the complete edge. However, a voxel that enclose an edge which connects arbitrary vertices only exists in a multi-resolution space, e.g. an octree. At least the root voxel contains the edge, as it contains all vertices in $\mathcal{V}$ and only edges between these are allowed. Moreover, this approach neutralizes the fast access that is accomplished by a spatial data structure, as many edges must be put into top-level voxel and thus have to be tested in each operation. Not only long edges are stored in a top-level voxel, but short edges that have an unfavorable location as well, e.g. if the end points of an edge are located in two voxels that are separated up to the root level.
However, it is also not desirable to store an edge in every leaf voxel that partially contains this edge, as this would significantly increase the cost of inserting and erasing edges. Consequently, an edge e is stored implicitly as attachment of its end points. In detail, every vertex $\mathbf{v}$ holds a list of attached edges $\mathcal{E}(\mathbf{v})$, corresponding to the mesh data structure presented in Section 3.1.

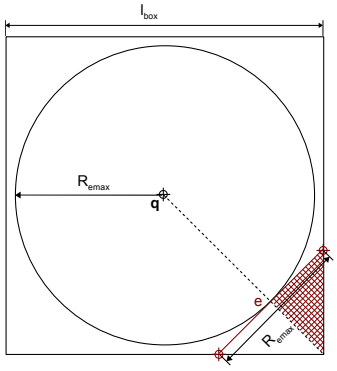


Figure 4.8: Relation between query of voxels and edge neighborhood: The size l_{box} of the bounding box for the voxel query must be large enough w.r.t. the neighborhood radius or maximum edge length $R_{e_{\max}}$ The shown edge e represents the boarder case of Equation (4.7). It is the line segment of length $\| e\| = R_{e_{\max}}$ and perpendicular to the diagonal of the bounding box with the edge points on the boarder of the bounding volume.

Insertion and Removal The insertion and removal of an edge $e = \overline{\mathbf{ab}}$ is performed by identifying its end points $\mathbf{a}, \mathbf{b}$ in the data structure and then inserting or removing the edge to or from the respective edge lists $\mathcal{E}(\mathbf{a})$ and $\mathcal{E}(\mathbf{b})$.

Neighborhood Query The query of an edge ball neighborhood $\mathcal{N}_R^{\mathcal{E}}(\mathbf{q}, \mathcal{M})$ is the search of all edges in the spherical volume centered at $\mathbf{q}$ and with radius R. The query is accelerated by identifying a suitable subset of voxels and then only testing the content of this subset, analogues to the query of vertex neighborhoods. Unlike in vertex neighborhood queries, the required voxel set is not determined by the smallest axis-aligned bounding box around the neighborhood sphere, as this bounding box must be large enough to guarantee that at least one end point of every edge that intersects with the spherical volume of the neighborhood is inside. In the mesh generation stage, addressed in Chapter 3.5, the maximum edge length and the radius of neighborhood queries is equal, i.e.

$$\max\{\|e\| \in \mathbb{R}_+ \mid e \in \mathcal{E}_{\mathcal{M}}\} \equiv R_{e_{\max}} \ .$$

The marginal case of an edge of length $R_{e_{\max}}$ that touches the sphere centered at $\mathbf{q}$ and with radius $R_{e_{\max}}$, but is located outside the bounding box with size l_{box}, is illustrated in Fig. 4.8. The shaded area forms an equal-sided triangle with an length of $\frac{R_{e_{\max}}}{2}$ for the two equal sides. Further, the sides can be described w.r.t. the diagonal of the bounding box. This results in

$$\frac{R_{e_{\max}}}{2} = \frac{\sqrt{2}}{2} l_{box} - R_{e_{\max}} \ .$$

Consequently, a complete search can only be guaranteed by relating the size of the bounding box l_{box} to the maximum edge length $R_{e_{\max}}$,

$$l_{box} \geq \frac{3}{\sqrt{2}} R_{e_{\max}} \ . \tag{4.7}$$

Let $\mathcal{D}_{\mathcal{NE}}$ denote the set of non-empty voxels inside the bounding volume of size l_{box} and centered at $\mathbf{q}$, with

$$\mathcal{D}_{\mathcal{NE}} = \{\mathbf{i}(\mathbf{q} - \frac{l_{box}}{2}), \ldots, \mathbf{i}(\mathbf{q} + \frac{l_{box}}{2}) \in \mathcal{D}\} \ .$$

Further, let $\mathcal{E}_{\mathcal{D}_{\mathcal{NE}}}$ denote the set of edges that have at least one end point inside of $\mathcal{D}_{\mathcal{NE}}$

$$\mathcal{E}_{\mathcal{D}_{\mathcal{NE}}} = \{\mathcal{E}(\mathbf{v}) \mid \mathbf{v} \in \bigcup_{\mathbf{i} \in \mathcal{D}_{\mathcal{NE}}} \mathcal{V}_\mathbf{i}\} \ ,$$

with the set of corresponding vertices $\mathcal{V}_\mathbf{i}$ of the voxel $\mathbf{i}$. Consequently, Equation (3.4) that defines the edge ball neighborhood can be formulated w.r.t. $\mathcal{E}_{\mathcal{D}_{\mathcal{NE}}}$,

$$\mathcal{E}_R(\mathbf{q}, \mathcal{M}) := \{e \in \mathcal{E}_{\mathcal{D}_{\mathcal{NE}}} \mid d_e^2(e, \mathbf{q}) \leq R^2\} \ .$$

Dynamic Voxel Space

In the Dynamic Voxel Space with a direct voxel linking extension, the limitations of the bounding box directly affect the relation between voxel size l and neighborhood radius $R_{e_{\max}}$. Only the central voxel $\mathbf{i}(\mathbf{q})$ is searched at the query of $\mathcal{N}_R^\mathcal{E}(\mathbf{q}, \mathcal{M})$. Consequently, the voxel size l must satisfy

$$l \geq \frac{3}{2\sqrt{2}} R_{e_{\max}} \ . \tag{4.8}$$

Extendable Octree

The Extendable Octree has no equivalent restrictions, nevertheless, the coupling of the voxel size to the maximum neighborhood radius in Equation (4.8) guarantees that no more than 27 voxels have to be searched i.e.

$$|\mathcal{D}_\mathcal{N}| \leq 27 \ .$$

4.2.3 Implementation Issues

Implementation considerations for the introduced data structures w.r.t. streaming surface reconstruction are provided in the following. Generally, computation can be accelerated by storing more information for every point, vertex or edge. Thus, the trade-off between memory consumption and computation time has to be optimized for the respective application, considering the available memory. In the following, additional acceleration considerations are shown that are used in this work.

Storage of Point Neighborhoods

The normal update step in the normal estimation stage requires a continuous update of point neighborhoods. During the insertion of a new sample point, all influenced points are identified, the respective neighborhoods are updated with the new point, and the surface normals are re-estimated. Hence, it is beneficial to store the respective neighborhood for each point, since thereby only the neighborhood of the new point has to be identified. The cost for this extension is an additional list that contains up to k links to neighboring points with the maximum neighborhood size k of the k-in-R neighborhood.
However, for the set of vertices in the mesh generation stage, this extension is not applicapble, because the vertex neighborhood is only searched once per point, as described in Section 3.5.

Avoiding Query Costs for Edge Insertion and Removal

Generally, the insertion and removal of edges requires a query of the voxel space. However, edges are only inserted or removed during the triangulation step. Since a query of the vertex and the edge neighborhood has been performed in the preceding step of candidate selection, a further query of voxels can be avoided by storing the corresponding voxel for every vertex and edge in the neighborhood during the triangulation update operation.

4.3 Summary and Discussion

In this Chapter, the problems of storage of and fast access to dynamically growing point sets and changing meshes, as required for the streaming surface reconstruction, are discussed. The problem is approached by voxelization of the 3D space and by providing a fast access to the discrete volume elements. Two possible implementations, the Dynamic Voxel Space and the Extendable Octree, are introduced and the application in streaming surface reconstruction is detailed.
The Dynamic Voxel Space is a single-resolution space. Its primary advantage is the small memory overhead. Only non-empty voxels have to be stored, with only a small per-voxel overhead for the tree structure. The access to the structure depends on the number of elements (voxels) in the tree. A drawback of this structure is that the spatial ordering is not represented, as the space is mapped to a planar search tree. An additional acceleration of the query is achieved by linking direct neighboring voxels during the creation and thus restoring spatial ordering information.
The Extendable Octree, however, is a multi-resolution space and preserves the spatial order of the voxels and the stored data. The memory consumption is potentially higher, since not only the leaf voxels have to be allocated, but the hierarchy above them as well. An advantage of the octree is the possibility of searching all voxels inside a discrete volume in a single traversal.
A comparison of the complexities of both data structures is hardly possible. The computational effort in a Dynamic Voxel Space is directly coupled to the number of non-empty voxels and consequently to the voxel size. Contrary, the Extendable Octree depends on the spatial resolution (i.e. the voxel size) and the compactness of the data. However, the levels of the Extendable Octree and consequently the complexity increase only by a $\log_8$-function with an increasing rectangular volume. The complexity of the Dynamic Voxel Space increases with a $\log_2$-function for a fully occupied voxel space. However, the real access is faster as the voxel space is not fully occupied, only non-empty voxels are stored and the stored data represents a surface, i.e. only a small part of a rectangular volume contains non-empty voxels. Hence, the average computational effort is similar for both data structures. If the data is compact, the Extendable Octree has a better performance. If the data is distributed e.g. a scene of many disconnected objects, the Dynamic Voxel Space provides a faster access.
Concluding, the choice of a suitable data structure for streaming surface reconstruction depends on the kind of object or scene that is to be reconstructed. In the context of manual scanning, objects are usually small, e.g. technical parts, pottery products or busts. Hence, the Extendable Octree is appropriate for access to the stored sample point sets and meshes. For sparse data sets with large expansion, the Dynamic Voxel Space is a better choice. In the applications in the following sections, the data is typically compact. Hence, the Extendable Octree is used.

5 Verification of Method

In this chapter, the *RT-SSR* method is verified using simulated scans on synthetic objects. A virtual scanner device that implements the sensor model from Chapter 2 is applied. The simulation setup is introduced and reconstruction results of selected scan sweeps along synthetic objects are shown. These example trajectories represent a group of typical real life situations occurring in manual scanning. Further, these results are analyzed and limitations of the method are identified. Moreover, the influence of the process parameters on the reconstruction method are discussed.

5.1 Simulations

For the evaluation of the streaming surface reconstruction method, scans of simple scenes consisting of primitive objects are generated and the true sample points and surface normals are stored. This allows for comparing the reconstruction results with ground truth data. With this approach, the normal estimation stage can be analyzed and verified.
In the following, the properties of the virtual scanner devices applied are described. Then, the synthetic scenes and generated scans are shown. Finally, results of the *RT-SSR* method are depicted.

5.1.1 Virtual Scanning Devices

A single-stripe scanner system with cylindrical geometry is used for validation. The cylindrical geometry implicates a central ray origin, thus single stripes may vary in sample density and accuracy. The geometric properties of the device are summarized in Tab. 5.1.
A limited sensor accuracy is simulated by applying zero-mean Gaussian noise to the virtual measurements. Four sensors with different noise characteristics are implemented: without noise (noiseless), noise with low standard deviation (low noise), noise with high standard deviation (high noise), and noise with a standard deviation that increases with the measured distance along a polynomial function (polynomial noise). In Fig. 5.1, the characteristic of the distance-dependent polynomial deviation is illustrated. The deviation parameters are summarized in Tab. 5.1. These different noise

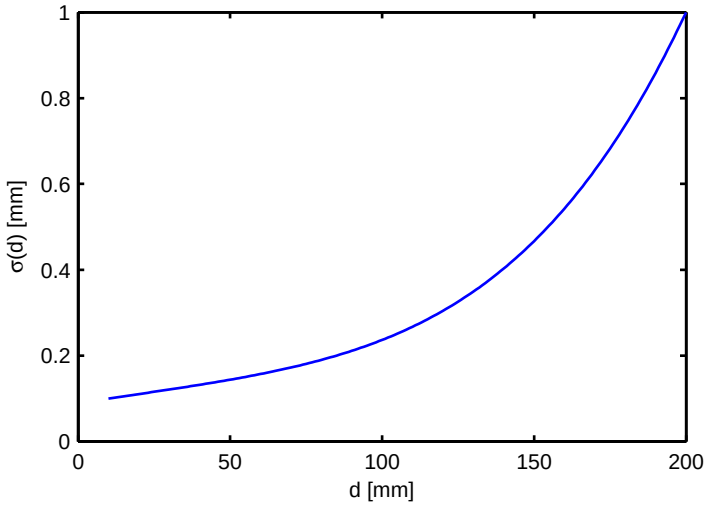


Figure 5.1: The characteristic curve of the distance-dependent polynomial deviation. The corresponding function is shown in Tab. 5.1.

Table 5.1: Attributes of the virtual range sensors: A single-stripe scanner system with cylindrical geometry is used for verification. The characteristic parameters have been discussed in Chapter 2. Four variants of the sensor are implemented, each with a different noise characteristic.

Geometric parameters:

N_u, N_v	u_0, v_0	$\Delta u, \Delta v$	$d_{\min}, d_{\max}$
1 px, 400 px	$-30°, 0$ mm	$0.15°, 0$ mm	10 mm, 200 mm

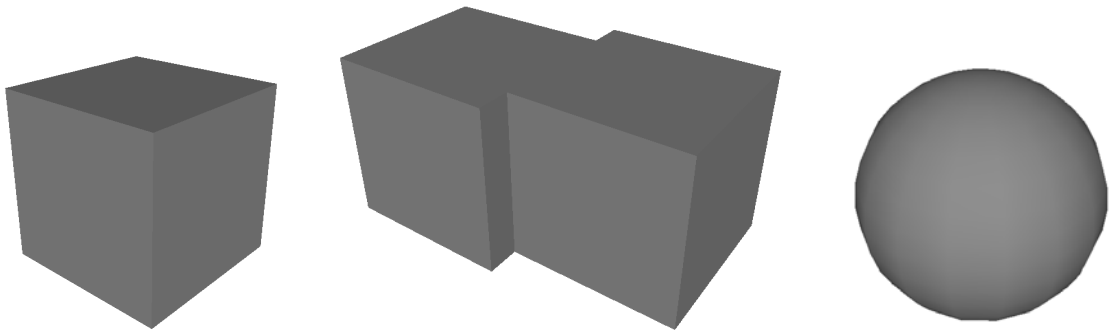
Noise parameters:

Sensor Variant	Shortcut	$\sigma(d)$
Noiseless	-N	$= 0$
Low Noise	-L	$= 0.1$
High Noise	-H	$= 0.8$
Poly. Noise	-P	$= 0.0897281 + 0.0010267\,d + 4.406e^{-10}d^2$

configurations can be used to verify the robustness of the reconstruction method w.r.t. sensor noise.
The range image at a particular pose is determined by calculating the intersection between the measurement ray and the object for every distance pixel. Moreover, the true sample point and surface normal are stored for every distance measurement.

5.1.2 Simulated Scenes and Scans

The virtual scenes consist of primitive analytic objects that allow for an accurate calculation of intersections with rays from a simulated scanner device. Cubes, spheres, planes, and combinations of these can be created within the simulation suite. In this section, three scenes are used: a single cube, two cubes, and a single sphere, as shown in Fig. 5.2.
The virtual scanner performs sweeps across the objects by defining virtual scan paths. In detail, these paths are defined by a set of key frames and the number of steps between two consecutive key frames. The scan poses are generated linearly interpolating between the key frames. Here, the interpolation method presented in Chapter 2.3 is used but with a fixed number of 100 steps between two consecutive key frames instead of a single interpolation time. Hence, each sweep contains $40,000$ sample points (see



(a) Scene 1: Single cube (b) Scene 2: Two cubes (c) Scene 3: Single sphere

Figure 5.2: Virtual scenes for verification of the streaming surface reconstruction: A single cube (a), two cubes with a shift in the x-direction (b), and a single sphere (c).

Tab. 5.1). Further, it is assumed that all distance pixel are measured synchronously and that the sensor pose is measured synchronously to the distance measurements. Hence, no further pose interpolation is required for each distance pixel.
The generated scans represent different situations and constellations: The scenes contain planar surfaces as well as curved regions and sharp edges. In the following, the scans are described and in Fig. 5.3 they are illustrated with the corresponding scan paths:

Scene 1 contains a single cube with an edge length of $100mm$, as shown in Fig. 5.2(a). The cube is used to analyze the behavior of the reconstruction method for planar areas as well as for sharp (convex) edges and corners. Two sweeps are generated:

- Scan *S1E* : Sweep across an edge consisting of 100 stripes with a scanner that looks upright onto the edge (Fig. 5.3(a))
- Scan *S1C* : Sweep across a corner consisting of 100 stripes with a scanner that looks upright onto the corner (Fig. 5.3(b))

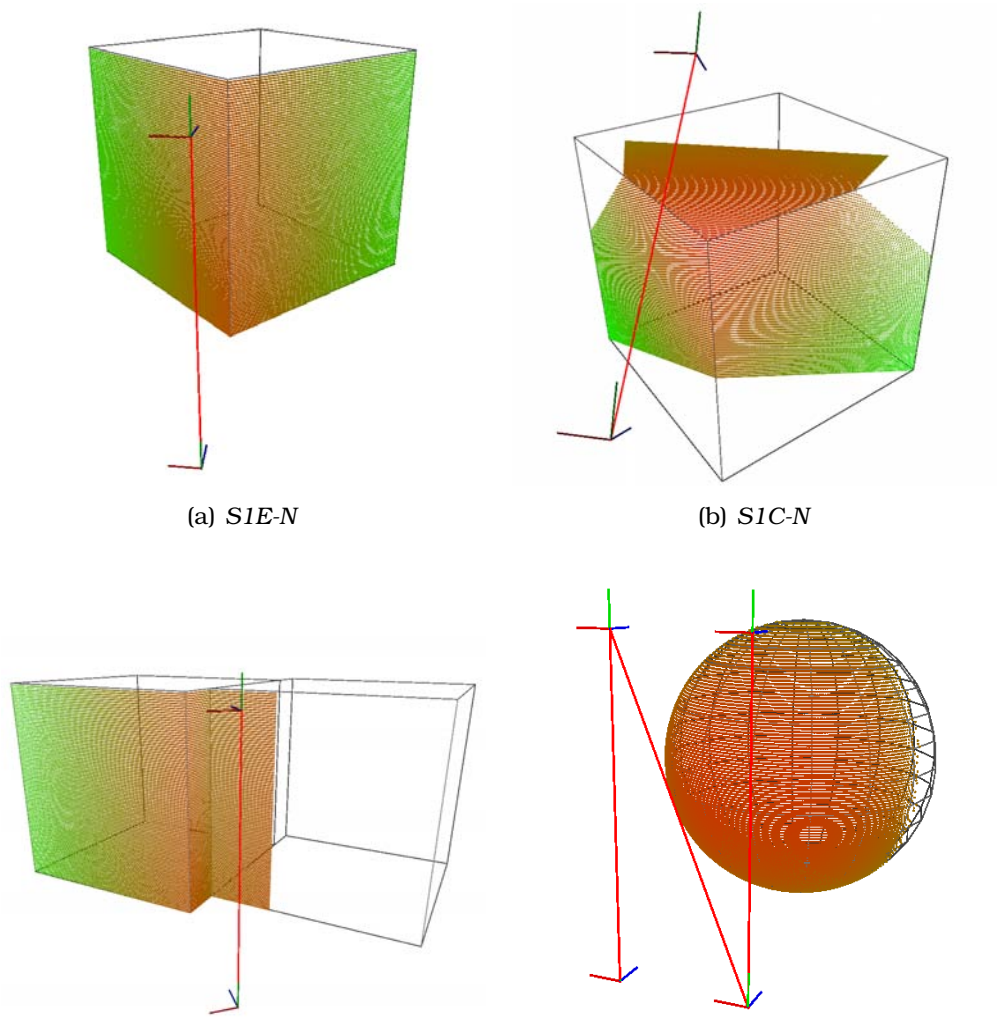
Scene 2 consists of the combination of two cubes, each with an edge length of $100\,mm$. One is shifted in x-direction by $20\,mm$, resulting in a concave edge. A scan that covers this edge is performed:

- Scan *S2* : Sweep into the concave edge with a scanner that looks into the inner edge (Fig. 5.3(c))

Scene 3 is a single sphere with a radius of $50\,mm$, centered at the origin. The sphere has a surface with constant curvature. A scan that consists of two overlapping sweeps is generated:

- Scan *S3* : Two overlapping sweeps with the scanner looking into the negative x-direction (Fig. 5.3(d))

All scenes are scanned using each of the four virtual sensors described in Tab. 5.1. In the following sections, the combination of scan path (*S1E*, *S1C*, *S2*, *S3*) and used sensor noise model (*-N*, *-L*, *-H*, *-P*) is encoded, e.g. the Scan *S1E* performed by a scanner with high noise is denoted by *S1E-H*.



(a) *S1E-N* (b) *S1C-N*

(c) *S2-N* (d) *S3-N*

Figure 5.3: Generated scans for the scenes in Fig. 5.2 without noise (*-N*): Scan *S1E* is a sweep across an edge of the cube in Scene 1 (a) and Scan *S1C* is the sweep across the corner of the cube (b). Scan *S2* is the scan of the concave edge in Scene 2 (c). In Scan *S3*, two overlapping sweeps across the sphere in Scene 3 are performed (d). The color of the sample points encodes their measured distance, the distances range from $10\,mm$ (red) to $200\,mm$ (green). Each sweep contains 100 stripes and thus $40,000$ sample points. The coordinate frames represent the location and orientation of the key frames, the red lines between the frames are the scan paths.

Table 5.2: Process parameters for the streaming surface reconstruction delivering the results presented in Fig. 5.4 and 5.5.

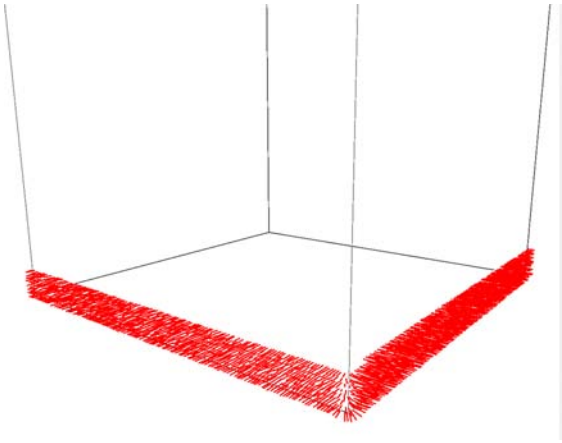
Parameter	**Symbol**	**Value**
Limitation radius	$R_{\min}$	$0.3\,mm$
Initial normal neighborhood	R_{n0}	$2\,mm$
Max. number of neighbors	k_n	20
Max. valid grazing angle	$\alpha_{s_{\max}}$	$80°$
Max. fast selection angle	$\alpha_{n_{\max}}$	$5°$
Min. number of selected neighbors	$n_{s_{\min}}$	5
Min. edge length	$R_{e_{\min}}$	$0.5\,mm$
Max. edge length	$R_{e_{\max}}$	$3\,mm$

5.1.3 Reconstruction Results

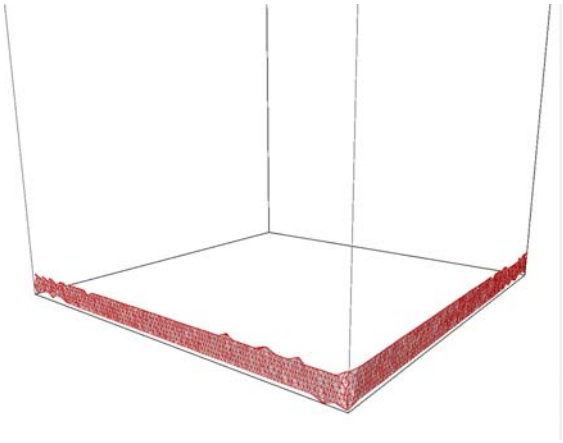
In the following, the reconstruction progress for the simulated scans is shown and processing times are listed. Results from the normal estimation stage and the mesh generation stage are presented. All scans are processed on a Dell Precision 390 system with an Intel Core2Duo 2.4GHz CPU, an NVIDIA FX3450 graphics adapter and 2GB memory, running a Linux operating system. The results are generated with a suitable set of process parameters, listed in Tab. 5.2. The choice and suitability of process parameters is further analyzed in the next section.

Reconstruction Progress In Fig. 5.4, the reconstruction progress is shown exemplarily for the processing of Scan *S1E-P*. Three snapshots of the processing illustrate the streaming character of the surface reconstruction. Fig. 5.4 shows the progress of both stages: the surface normals of the normal estimation (a,c,e) and the surfaces of the mesh generation (b,d,f). Scan *S1E-P* is a bottom-up sweep along the cube's edge. Hence, the data is continuously growing, starting at the bottom of the cube.
Comparing left and right, Fig. 5.4 shows that more estimated surface normals than vertices are visible in the mesh at a certain time. This delay is caused by the selection stage described in Chapter 3.4, as the upper-most stripes have an unbalanced neighborhood and are thus delayed by the selection criteria. Further, the continuous change of surface normals at the edge of the cube is visible. This effect is caused by the use of local neighborhoods for the estimation as described in Chapter 3.3.

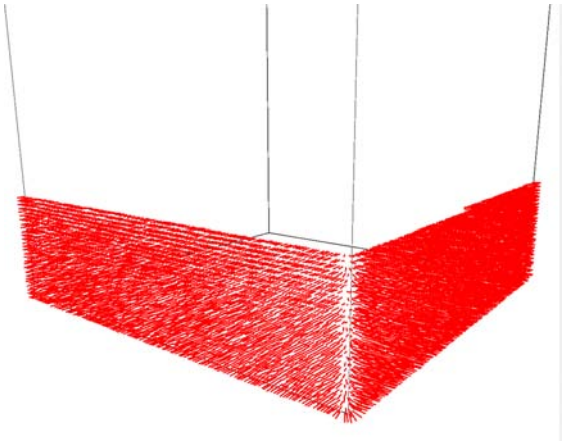
Reconstruction Results In Fig. 5.5, the processing results for the Scans *S1C-P*, *S2-P*, and *S3-P* are shown. The corresponding statistics are summarized in Tab. 5.3. number of input points, number of reduced points after the limitation step, and amount of vertices and triangles in the final mesh. Further, the overall processing time is measured. The results in Fig. 5.5 show that the method creates good results in various situations, including scenes containing sharp edges and corners (a), concave parts (b), and surfaces with constant curvature (c).



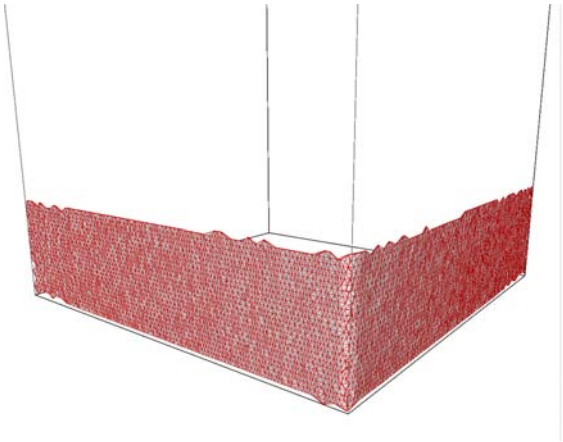
(a) after 3000 inserted points



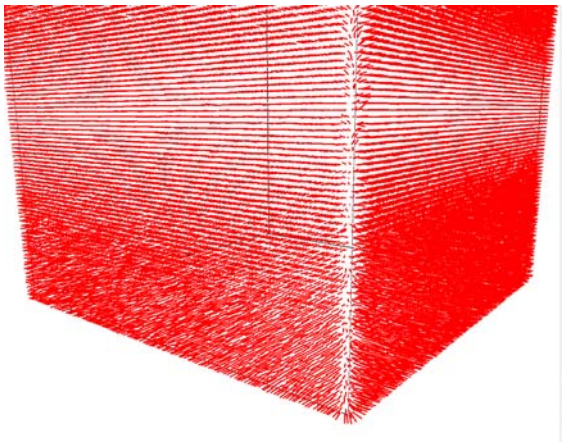
(b) after 3000 inserted points



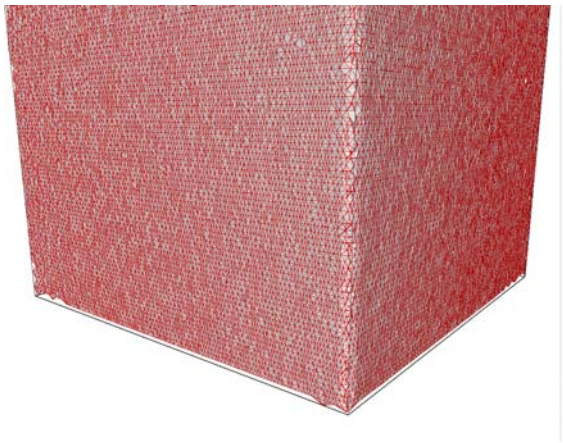
(c) after 10000 inserted points



(d) after 10000 inserted points



(e) complete model

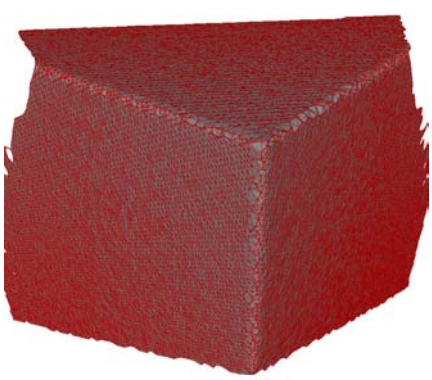


(f) complete model

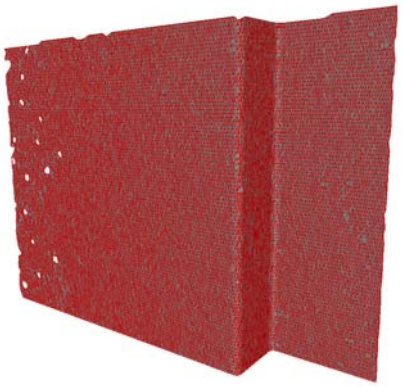
Figure 5.4: Progress of mesh generation for Scan *S1E-N*: Results from Scan *S1E* after the normal estimation stage (a,c,e) and results after the mesh generation stage (b,d,f).

Table 5.3: Results of surface reconstruction for the simulated scans in Fig. 5.5: Number of input points, number of reduced points after limitation, amount of vertices and triangles in the final mesh, and overall processing time.

Scan	Input Pts.	Reduced Pts.	Vertices/Triangles	Time [s]
S1E-P	35500	29639	20646/61411	2,649
S1C-P	36631	27271	17828/53083	2,668
S2-P	37800	27694	16678/49617	2,539
S3-P	53323	21627	11265/33509	2,686



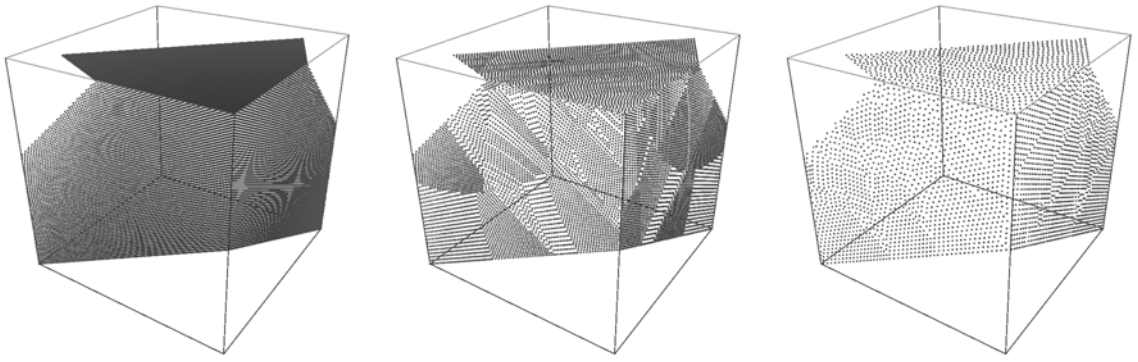
(a) *S1C-P*



(b) *S2-P*

(c) *S3-P*

Figure 5.5: Surface Reconstruction results for Scans *S1C-P*, *S2-P*, and *S3-P* using the parameters in Tab. 5.1.



(a) $R_{\min} = 0\,mm$ (b) $R_{\min} = 1\,mm$ (c) $R_{\min} = 2\,mm$

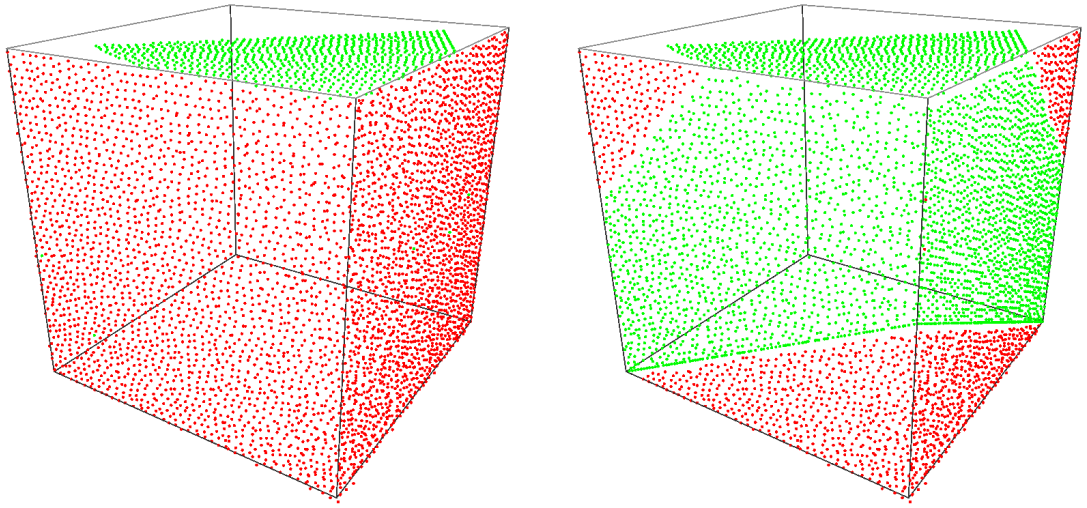
Figure 5.6: Density limitation with different radii: Figure (a) shows the sample point set of Scan *S1C-N* without limitation. The corresponding sets after limitation with $R_{\min} = 1\,mm$ (b) and $R_{\min} = 2\,mm$ (c) show variations in the point density due to the relation between native sample density of the input data and limitation radius.

5.2 Analysis and Discussion

In the previous section, the progress of streaming surface reconstruction and its results have been visualized for a selection of simulated scans and a suitable set of process parameters. In this section, the influence of the process parameters on normal estimation and mesh generation are analyzed and limitations to the method are discussed.

5.2.1 Density Limitation

The density limitation defines the maximum number of points within a finite volume. Thus, the sample density on the scanned surface is limited. In Section 3.2, two possible limitation methods have been introduced: the simple limitation and the replacement limitation.
For both methods, the maximum sample density is controlled by the limitation radius $R_{\min}$. Fig. 5.6 shows the limited point sets for three different limitation radii $R_{\min} = 0\,mm$, $R_{\min} = 1\,mm$, and $R_{\min} = 2\,mm$. Variations in the point density are visible, caused by the relation between native sample density of the input data and limitation radius. In detail, the sample density of Scan *S1C* decreases along the cube's faces. A distance smaller than $2R_{\min}$ between two adjacent points results in gaps that are not closable by further points, as described in Chapter 3.2 and illustrated in Fig. 3.3.v However, this variance of the point density is considered in the subsequent normal estimation step and thus its impact on the surface reconstruction is negligible. The mesh generation stage compensates this local high densities with its own simple limitation step.
In Fig. 5.7, the result of inserting first Scan *S1E-H* (high noise) and then Scan *S1C-L* (low noise) is shown for the simple limitation (a) and for the replacement limitation method (b), both with a limitation radius of $R_{\min} = 2\,mm$. With the simple limitation, points with low noise (green) are rejected, if nearby points with high noise (red) are inserted earlier. The replacement limitation overcomes this problem. However, it generates a high local point density and a gap at the border of Scan *S1C-L*. This effect originates from the replacement strategy itself, as noisy points inserted earlier define the anchor points. If a point's distance to the border of a point from the second scan is smaller than R_{min}, it is shifted towards the points with higher quality. However,



(a) Simple limitation (b) Replacement limitation

Figure 5.7: Comparison of simple limitation and replacement limitation by first inserting Scan *S1E-H* and then Scan *S1C-L*, both with a limitation radius of $R_{\min} = 2\,mm$: The simple limitation (a) prefers points with high noise points (red) that are inserted earlier over points with low noise (green). The replacement limitation overcomes the input order dependency, however, it generates a local high point density and a gap at the border of the low noise scan (b).

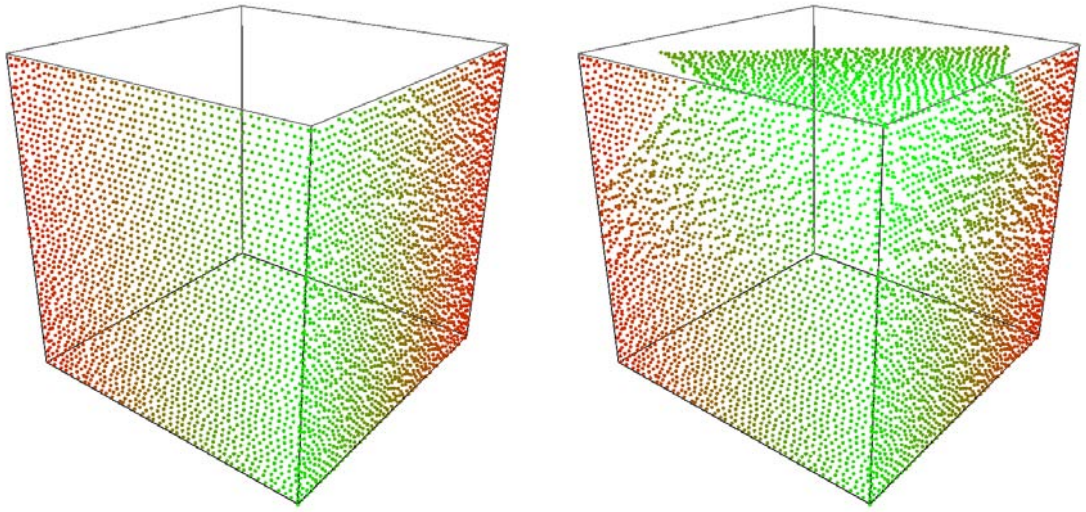
this shift is limited to a distance of R_{min} from the corresponding anchor point. If two point sets with continuously decreasing qualities are combined, the shifting effect is less distinct. This is exemplarily shown in Fig. 5.8, where the Scans *S1E-P* and *S1C-P* (both with polygonal noise) are combined for a limitation radius of $R_{\min} = 2\,mm$.

5.2.2 Normal Estimation and Selection

The estimation of a point's surface normal is performed by fitting a least square plane through a k-in-R neighborhood (see Chapter 3). Correctness and accuracy of normal estimation depend on the neighborhood and the local sampling. In the following, the impact of process parameters on the estimation and the verification (selection) are analyzed.

Normal Estimation The neighborhood of each point is controlled by the limitation radius $R_{\min}$, the initial neighborhood radius R_{n0}, and the maximum number of points in the neighborhood k_n.
The initial radius R_{n0} defines the maximum search radius for possible neighboring points and thus controls the required sample density that enables the selection of the point. A larger radius allows for the estimation of valid surface normals from sparsely sampled surfaces. However, an increasing radius also increases the computational effort for searching the initial ball neighborhood. This aspect is further discussed in Section 5.2.4. A minimal requirement for R_{n0} w.r.t. to $R_{\min}$ and k_n is given by Equation (3.13). As in the context of visual feedback, the delay caused by a possible rejection in the selection stage should be low, the initial neighborhood radius R_{n0} is adapted for each point w.r.t. the expected sample density. Let $\mathbf{q}$ denote a newly inserted sampled point. In this work, its initial neighborhood radius is determined by

$$R_{n0}(\mathbf{q}) = \max\{R_{n_{\min}}, 2\,\delta_{\mathbf{q}}, R_{n_{user}}\}\ .$$



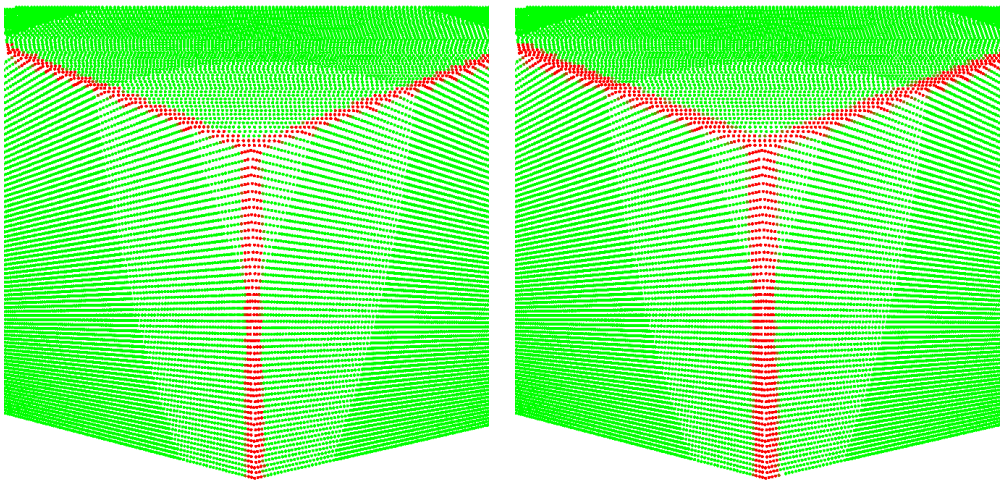
(a) Processing of Scan *S1E-P* (b) Processing of Scans *S1E-P* and *S1C-P*

Figure 5.8: Replacement limitation for two scans with similar noise and a limitation radius of $R_{\min} = 2\,mm$: First, Scan *S1E-P* is inserted (a), then Scan *S1C-P* is added (b). The effect of local high densities at the borders of Scan *S1C-P* are less distinct as in the results shown in Fig. 5.7(b).

This strategy guarantees the minimum radius $R_{n_{\min}}$ and compensates an decreasing sample density. The manual override $R_{n_{user}}$ can be used to define a larger minimum radius than $R_{n_{\min}}$. This override is particularly important for 1D range sensors (e.g. light-stripe sensors), as the movement of such a sensor potentially defines the distance between two subsequent acquired stripes and thus the sample density in this movement direction.

The parameters $R_{\min}$ and k_n control the extension of the point neighborhood. An increasing limitation radius $R_{\min}$ results in decreasing sample density, as already discussed in Section 5.2.1. Thus, the area on the surface of a neighborhood that should contain k_n points increases with an growing value of $R_{\min}$. An increasing number of points k_n at a constant $R_{\min}$ also increases the area on the surface. Hence, a large limitation radius or number of points result in a strong smoothing of the points and a possible loss of geometrical details due to this smoothing. Fig. 5.9 illustrates the estimated surface normals at the edges and the corner of a cube (Scan *S1C*) with two different maximum number of points $k_n \in \{15, 30\}$ and a constant radius $R_{\min} = 0.5\,mm$.

The maximum number of points in the neighborhood k_n is set in order to decrease the neighborhood after the initial calculation and to thus limit the computational effort for the estimation. It directly influences the robustness of the estimation (see Equation (3.9)). A small value for k_n w.r.t. the sample density can result in an unstable normal estimation and thus in a rejection of points at the selection stage. In Fig. 5.10, the rejection of points due to a too small value of k is illustrated for the example of a low density in sweep direction. Concluding, a smaller limitation radius $R_{\min}$ requires either a more dense sampling of the surface or a higher value of k_n. For any combination of $R_{\min}$ and k_n, the initial neighborhood radius R_{n0} has to be chosen properly, i.e. it must guarantee that k_n points can be sampled with the chosen value of $R_{\min}$.



(a) $k_n = 15$ (b) $k_n = 30$

Figure 5.9: Influence of $R_{\min}$ and k_n on the normal estimation: The sample points of Scan *S1C* without sensor noise are colorized w.r.t. the difference between the corresponding estimated surface normal and the true surface normal. The difference ranges from $0°$ (green) to $> 10°$ (red). The regions along the edges differ from the true normal, as the latter rotates by $90°$ at the edge and the estimated surface normal changes continuously (see Chapter 3.3). A large value of k_n at a constant $R_{\min} = 0.3\,mm$ increases the region of differing surface normals, i.e. a higher value of k increases the smoothing.

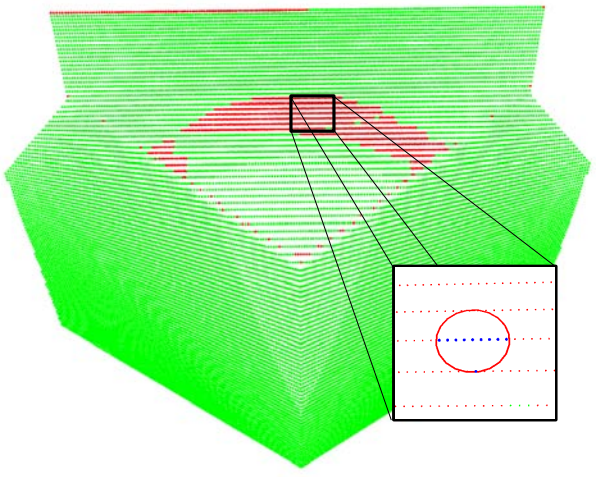


Figure 5.10: Rejection of points due to a too small value of k_n: Sample points of Scan *S1C* after surface normal estimation that have passed the selection stage (green) or have been rejected (red). The rejection is caused by a too small value of $k_n = 10$ (for $R_{\min} = 0.3$) w.r.t. the distance of adjacent stripes on the surface, as shown in the zoomed detail. The neighborhood only contains points of a single stripe, plus one point of an adjacent stripe, as the radius is too low.

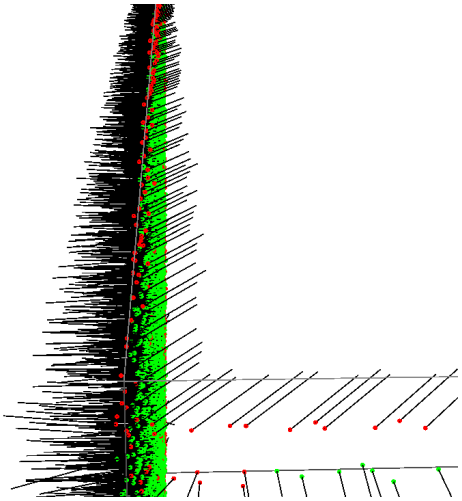
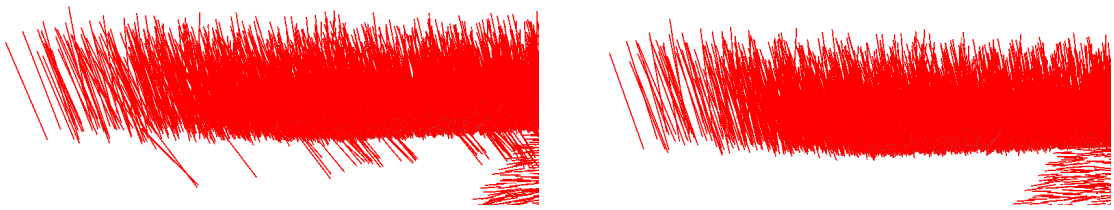


Figure 5.11: Rejection of incorrect surface normals: Several surface normals at the border of the point set are estimated incorrectly (i.e. they point into the wrong direction) due to sparse neighborhoods. The points are not selected (selected points are green, rejected are red) due to a rejection by the criteria $\mathfrak{c}_1$ to $\mathfrak{c}_3$.



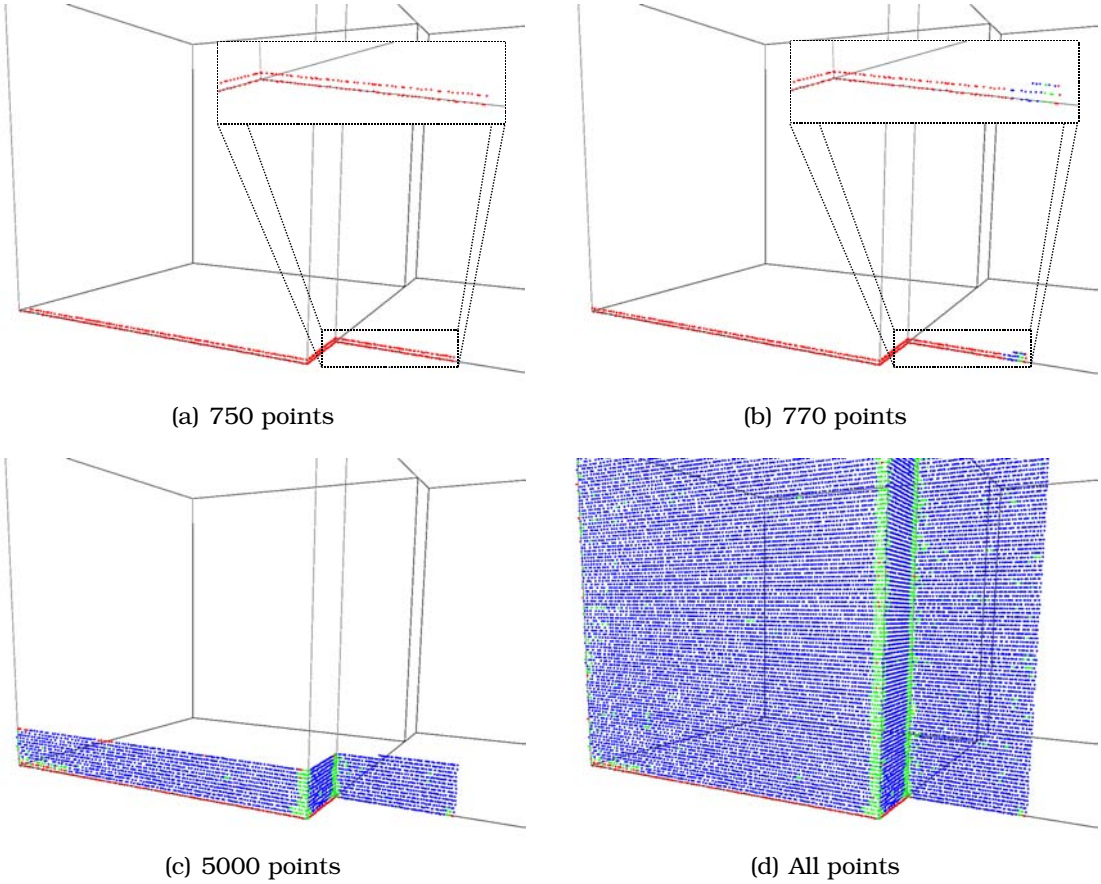
(a) $\alpha_{s_{\max}} = 90°$ (b) $\alpha_{s_{\max}} = 70°$

Figure 5.12: Flipped surface normals caused by a flat line-of-sight onto the surface for Scan *S2-H* and the parameter set: $R_{\min} = 0.3\,mm$, $k_n = 15$, $R_{n0} = 2\,mm$. Without the $\mathfrak{c}_0$-criterion ($\alpha_{s_{\max}} = 90°$), some selected surface normals point inside the volume, with $\alpha_{s_{\max}} = 70°$ this error are avoided.

Selection The local sampling of a manual scanner system depends on the operator and the movement of the scanning device. However, the selection step verifies the sampling density and rejects undersampled areas and unbalanced local neighborhoods. In Fig. 5.11, an example for incorrect surface normals caused by sparse neighborhoods at the border of the scan is shown. The verification step filters outliers and other erroneous measurements: these points either have only few points in the neighborhood or the neighboring points are randomly scattered. Both cases are rejected by the selection criteria $\mathfrak{c}_1$ to $\mathfrak{c}_3$, as defined in Chapter 3.4. They require a neighborhood that contains sufficient number of points and is homogeneously distributed along the surface, not perpendicular to it.

The $\mathfrak{c}_0$-criterion defining the maximum valid grazing angle $\alpha_{s_{\max}}$ avoids that samples with a flat line-of-sight onto the surface in combination with noisy measurements cause wrong decisions on the sign of the estimated surface normal, as explained in Chapter 3.4. In Fig. 5.12, the selected surface normals generated from Scan *S2-H* are shown with ($\alpha_{s_{\max}} = 70°$) and without ($\alpha_{s_{\max}} = 90°$) the validation of the grazing angle. Here, the noise in association with the chosen estimation parameters ($R_{\min} = 0.3\,mm$, $k_n = 15$, $R_{n0} = 2\,mm$) causes inaccurate estimations of normal directions. Without the $\mathfrak{c}_0$-verification, some selected surface normals point inside the volume; applying the verification, this error is avoided.

During the selection, the criteria $\mathfrak{c}_0$ to $\mathfrak{c}_3$ are very strict and sometimes points with



(a) 750 points (b) 770 points

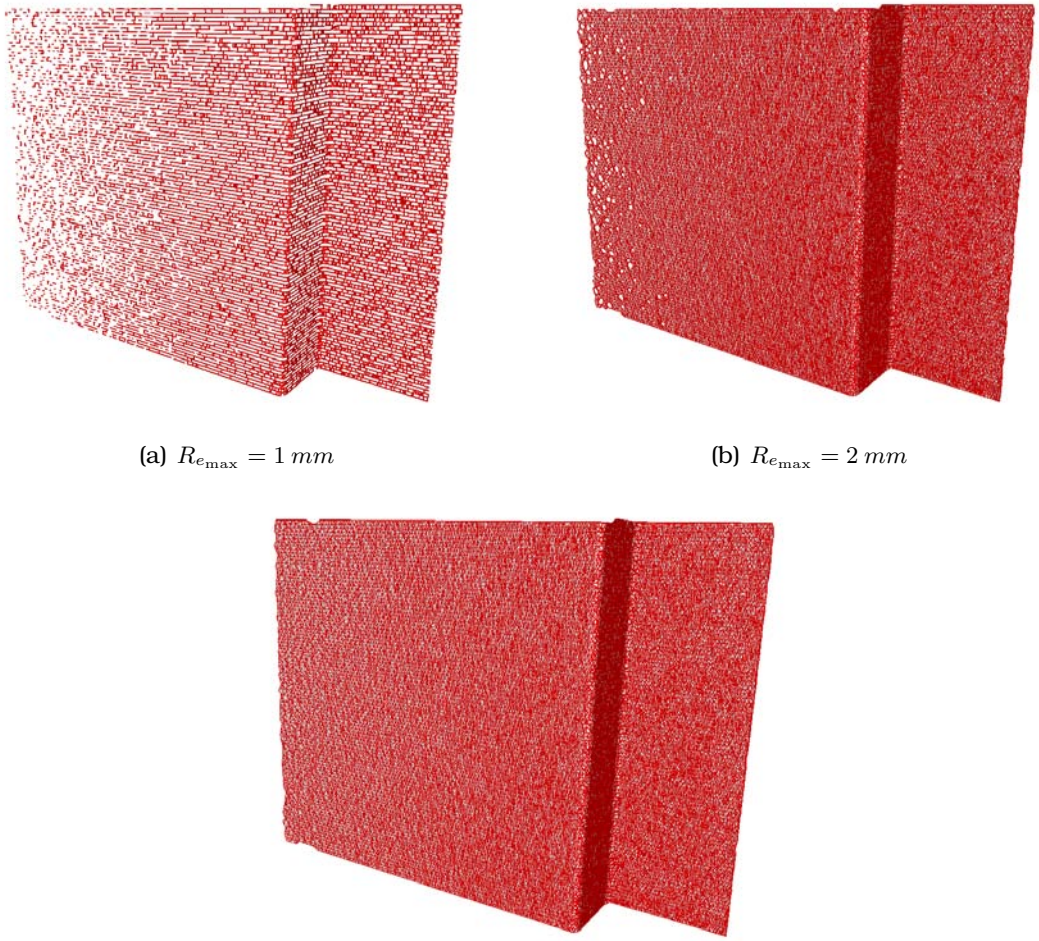
(c) 5000 points (d) All points

Figure 5.13: Interaction between regular selection and fast selection during processing of Scan *S2-P*: Rejected points are marked red, regularly selected points are green, and fast selection points are blue. The fast selection requires $n_{s_{\min}} = 5$ points and an angle smaller than $\alpha_{n_{\max}} = 5°$. After 750 insertions, nothing is selected (a). The insertion of 20 more points results in a regular selection of 5 points and the fast selection of several nearby points (b). The fast selection strategy propagates itself and results in a rapid point selection in flat areas, whereas curved areas (e.g. the edges) have to be selected regularly (c,d), because the surface normals change too much.

correct surface normals are rejected. The fast selection criterion $\mathfrak{c}_{fast}$ is designed to detect and select these rejected points by comparing estimated surface normal to those of selected points nearby. It is parametrized by the minimum number of selected neighbors $n_{s_{\min}}$ and the maximum difference angle $\alpha_{n_{\max}}$ between the examined surface normal and the average of those of the selected neighbors. In Fig. 5.13, the interaction between regular selection and fast selection is illustrated. Here, the fast selection is parameterized with $n_{s_{\min}} = 5$ and $\alpha_{n_{\max}} = 5°$, i.e. only small differences are allowed. The fast selection strategy propagates itself and results in a rapid point selection in flat areas, whereas curved areas (e.g. the edges) have to be selected regular, because the surface normals change too much.

5.2.3 Mesh Generation

The mesh generation stage is parametrized by two parameters: the minimum edge length $R_{e_{\min}}$ and maximum edge length $R_{e_{\max}}$. The minimum edge length is the radius of the second limitation stage and directly controls the resolution of the edge, as presented in Chapter 3.5. The maximum edge length defines the maximum distance between two vertices that can be connected, larger distances between vertices result in holes or unconnected areas. The two parameters are coupled, as the neighborhood



(a) $R_{e_{\max}} = 1\,mm$ (b) $R_{e_{\max}} = 2\,mm$

(c) $R_{e_{\max}} = 3\,mm$

Figure 5.14: Coupling between $R_{e_{\min}}$ and $R_{e_{\max}}$ during mesh generation from Scan *S1C* for different values of $R_{e_{\max}}$ and a constant $R_{e_{\min}} = 0.2mm$: A low value of $R_{e_{\max}}$ in relation to $R_{e_{\min}}$ results in an incomplete reconstruction (a). Increasing the radius $R_{e_{\max}}$ results in more candidate points for triangulation, and thus in an improved triangulation result (b). A further increase of $R_{e_{\max}}$ allows for triangulation of more sparsely sampled areas, e.g. the border areas (c).

volume must at least encompass the closest possible vertices around the examined vertex. The limitation results in point distances between $R_{e_{\min}}$ and $2\,R_{e_{\min}}$ (see Chapter 3.2). Hence, the maximum edge length must at least satisfy

$$R_{e_{\max}} > 2\,R_{e_{\min}} \quad . \tag{5.1}$$

However, a large value for $R_{e_{\max}}$ unnecessarily increases the computational effort. In Fig. 5.14, the coupling between $R_{e_{\min}}$ and $R_{e_{\max}}$ is illustrated for $R_{e_{\max}} \in \{1, 2, 3\}mm$ at a constant $R_{e_{\min}} = 0.2\,mm$. The small radius $R_{e_{\max}} = 1\,mm$ results in a poor triangulation due to incomplete sets of candidate points. The medium radius $R_{e_{\max}} = 2mm$ generates a sufficient amount of candidate points at each update, the large radius $R_{e_{\max}} = 3mm$ even allows for closing larger distances between the vertices, i.e. more sparsely sampled areas are also triangulated. In this work, a factor of at least three between both radii is used, as this trade-off has shown good results.

5.2.4 Voxel Size and Computation Time

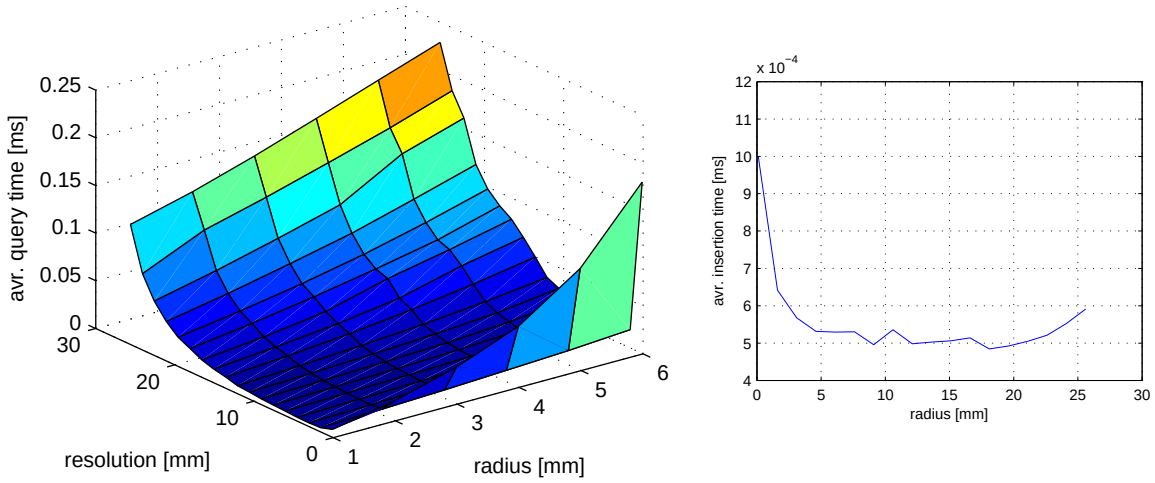
The voxel size l_0 controls the voxelization of the space, i.e. the amount of voxels generated for a certain extension of the data set. Hence, it controls the computational effort for insertions and neighborhood queries on point sets and meshes, as described in Chapter 4.
A neighborhood query is composed of the search of all voxels inside the neighborhood sphere and of the test of all points inside each voxel, as described in Chapter 4.2. In the *RT-SSR* method, the number of points in any finite volume is limited by the density limitation technique, as presented in Chapter 3.2. Hence, the maximum amount of points per voxel increases with the voxel size. A very small space resolution results in high computational effort due to a very fine voxelization, i.e. a large amount of voxel and thus a number levels in the octree. However, a very large space resolution also causes high computational effort: while only few voxel must be traversed, they cover a large area and potentially contain a large amount of points. Fig. 5.15(a) illustrates this characteristics of the computation time for ball neighborhood queries with different space resolutions and neighborhood radii using the example of the points of Scan *S2-N*. The results confirm the expected development. Moreover, it shows that there exists a flat minimum for resolutions w.r.t. the neighborhood radius R that lies in the range

$$2R \leq l_0 \leq 5R \ ,$$

w.r.t. the neighborhood radius R.
The insertion of a point principally only requires the query of the corresponding voxel, as described in Chapter 4. This would implicate that the computational effort decreases with the amount of voxel, i.e. with an increasing voxel size. However, a denisty limitation is applied during insertion, as described in Chapter 3.2, and requires an additionally query of the neighborhood with radius $R_{\min}$. Thus, the computational effort for the insertion increases also for high voxel sizes. Fig. 5.15(b) shows the average times for inserting all points of Scan *S2-N* at different space resolutions. The results confirm the assumed development of the computation time for insertion operations. Measurements with different data sets show similar results for both, query and insertion.
During surface reconstruction, more queries then insertions are performed. Consequently, the resolution is optimized for queries. In this work, the space resolution is set to

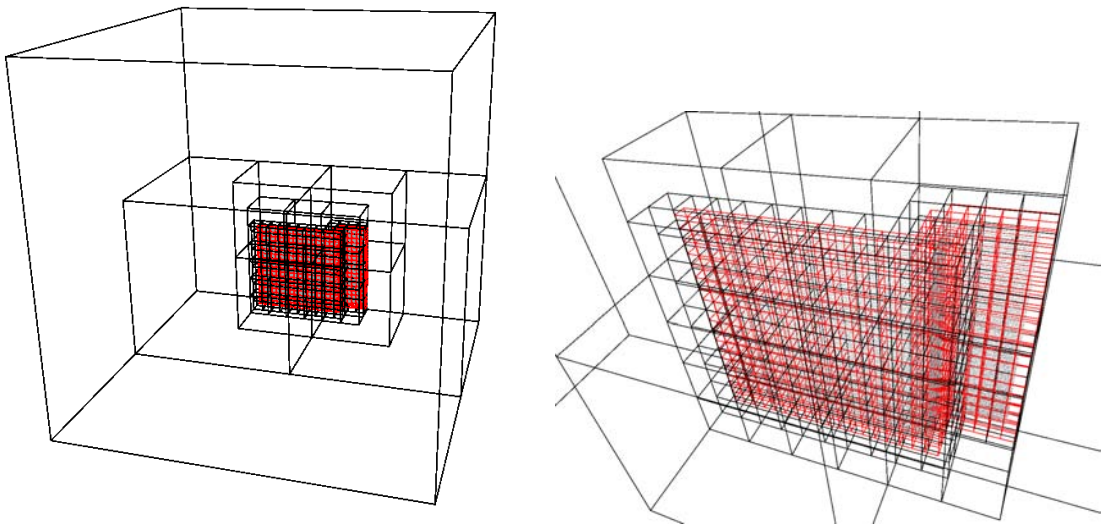
$$l_0 := 3\,R \ ,$$

as this guarantees that the computation time is at or near the minimum time. This coarse voxelization results always in a small amount of voxel w.r.t. the point density and mesh resolution, generating only a small memory overhead, even for larger objects. In Fig. 5.16, the octree for the points of Scan *S2-N* at a space resolution of $l_0 = 3\,mm$ is shown as an example.
The computation time of the *RT-SSR* method for a certain set of process parameters increases slowly by a logarithmic function with the insertion of more points, as the global operations require more time. However, the computation time of local operations in the different stages depend on the choice of the process parameters.
The calculation of mean and covariance in the normal estimation stage have a linear complexity w.r.t. the number of points in the neighborhood. This is controlled by the parameter k_n, as it is the maximum number of points in the neighborhood.
The triangulation update requires the mapping of all n vertices and m edges in the neighborhood, which has a linear complexity. Further, the candidate vertices must be



(a) Query time (b) Insertion time

Figure 5.15: Computational effort for neighborhood queries: The average time for insertion (a) and query (b) of the points from Scan *S2-P* at different space resolutions are visualized. Both operations are performed with a limitation radius of $R_{\min} = 0.2\ mm$. The insertion time potentially decreases with higher resolutions. However, very large resolutions causes an increasing effort, caused by the limitation. The query time increases with higher neighborhood radii and has a flat minimum for space resolutions that are three to five times the neighborhood radius, as a small value for the space resolution causes a high effort for the voxel query and a very high value results in a high amount of points per voxel.



(a) (b)

Figure 5.16: Voxelization on Scan *S2-N*: The complete octree (a) and the close-up of the area with the scan (b) for a space resolution of $l_0 = 6\ mm$.

sorted, witch takes O($n \log n$) steps. Finally, all candidates must be tested with every neighboring edge in the worst case, resulting in O($n\,m$) steps.
Consequently, the number of points, vertices, and edges in the respective ball neighborhood should be only as large as necessary for a robust surface reconstruction, as described in the previous sections. The responsible parameters are the maximum number of points k_n for the normal estimation and the ration between $R_{e_{\min}}$ and $R_{e_{\max}}$ for the mesh generation.

5.3 Summary

This chapter analyzes the *RT-SSR* method using simulated scans on virtual scenes. The selected scenes represent real world situtations and are applied to verify the method. Moreover, different noise models are added to the simulated distance measurements and to test the algorithm's robustness.
The method shows correct results for all examined data if a suitable set of parameters is chosen, even for noisy data. The correct choice of parameters proves to be essential, as the parameters are coupled and the selection of inappropriate parameters causes bad reconstruction results or high computational effort.
As expected, using density limitation with replacement overcomes the rejection of high quality points, as observed when applying the simple limitation. However, it can generate local high densities when combining scans with very different quality values, yet the effect on the subsequent normal estimation is minor. The normal estimation is controlled by the limitation radius, the initial neighborhood radius, and the maximum number of points. The alteration of one parameter typically requires an adaption of the other two, as the three are coupled. Here, limitation radius and maximum number of points control the expansion of the neighborhood. A small radius delivers a sparse or unbalanced neighborhood, resulting in a rejection of the sample points during the selection stage. A large radius causes a strong smoothing of coordinate and surface normal. The initial neighborhood radius controls the required sample density for at least a coarse estimation of surface normals, and thus affects the delay of the reconstruction process.
The mesh generation stage is parametrized by the minimum and maximum edge length. The maximum edge length is the neighborhood radius for the search of candidate points and nearby edges. It must be sufficiently higher than the minimum radius, as otherwise not enough candidate points can be detected in the neighborhood. However, the computation time increases with a higher ratio between minimum and maximum edge length. Thus, the maximum edge length should be chosen only as large as necessary for a robust mesh generation.
The computational effort for global operations on the stored data is coupled to the voxel size used. The computational cost of insertion and query has a single minimum for voxel sizes that are twice to fifth of the used neighborhood radius. Hence, a resolution that is the threefold of the radius is recommended in this work to guarantee a small computational effort.
In the following chapter, the applicability of the *RT-SSR* method as the core component of a visual feedback system for a hand-guided scanner system is demonstrated. Here, an optimized parameter set is used.

6
Manual Digitization

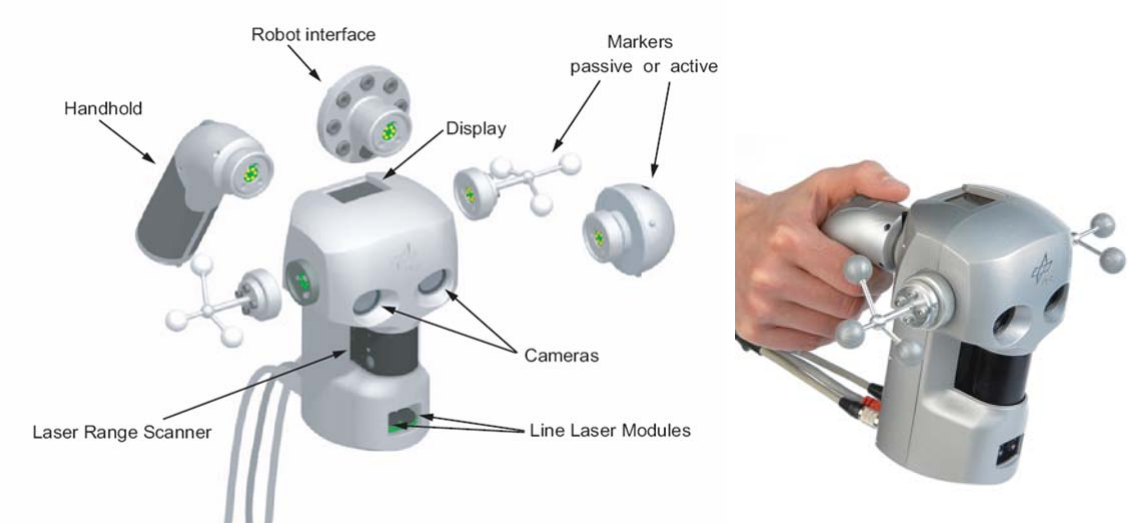
This Chapter presents a visual feedback application for a particular manual scanner system, the **DLR Multisensory 3D Modeler**. The system and its integrated range sensors as well as possible pose sensors are introduced. Further, the most suitable configuration of the *RT-SSR* process and the design of a visualization for this particular scanner system is discussed. Possible extensions to this visualization that improve the usability of the scanner system are introduced. Finally, scanning results and selected applications are presented.

6.1 The DLR Multisensory 3D Modeler System

Current commercial and research scanner systems are typically specialized, i.e. optimized for one or a limited number of applications. System specific attributes, e.g. working range or accuracy, as well as reliability of the results are optimized for a certain task. As an example, the modeling of objects requires high precision, while for autonomous robotic exploration tasks, a high reliability and large working range is desired. Further, manual scanner systems have to be small and light weighted.

At the DLR Institute for Robotics and Mechatronics, the challenge of a suitable sensor system for multiple applications in the field of computer vision and robotics has been solved by the development of a multi-purpose vision platform, the **DLR Multisensory 3D Modeler (3DMo)**. In Fig. 6.1, the system is shown. It integrates cameras and laser modules. Three different range measurement methods with different properties are implemented. The system is used for robotic exploration [Sup07], for 6-DoF object tracking [Sep08], and for object recognition and grasp planing [OEF+06]. In this work, the system is used for manual scanning of objects.

In the following, the hardware, communication, and synchronization of the 3DMo system are briefly described. A detailed description of the hardware design is found in the work of Suppa et al. [SKL+07], the communication concept is explained by Bodenmüller et al. [BSSH07]. Further, the range sensors, external pose measurement, and system calibration are introduced.



(a) Design drawing (b) Real system

Figure 6.1: The DLR Multisensory 3D Modeler: The design drawing (a) shows the central components and possible adapters for either robot-mounted operation or for hand-held applications. Further the real system in hand-held configuration is shown (b).

6.1.1 Hardware Components

The sensing components of the 3DMo are a pair of FireWire cameras, two line-laser modules, and a DLR laser-range scanner (LRS). The latter is an independent laser range sensor and is described by Hacker et al. [HDH97]. The integrated AVT Marlin cameras[1] have a resolution of 780×580 pixel and a maximum frame rate of 50Hz. Two different lenses with a focal length of $f = 6$mm and $f = 12$mm can be mounted. The base distance of the cameras is 50mm, which represents a trade-off between perspective difference and range precision assuming a general working range between 100mm and 2000mm. The line-laser modules have an opening angle of $60°$ and a wavelength of 635nm.

6.1.2 Communication and Synchronization

The 3DMo is connected to a PC system running a Linux operating system, the so-called **Sensor PC**, via a FireWire bus[2]. A protocol bridge application gathers the data from the hardware components and provides a concurrent, synchronized and event-driven access to the data.

The synchronization and data labeling is implemented w.r.t. the design considerations in Chapter 2.3. All incoming data is labeled with a unique timestamp. This timestamp generation is synchronized to a hardware pulse generator in the 3DMo. Further, the protocol bridge provides the labeled raw data to subsequent processing stages via a shared memory interface. This design allows for an implementation of subsequent processing in separate applications. This communication and synchronization concept has already been described by Bodenmüller et al. [BSSH07].

[1] Allied Vision Technologies (`http://www.alliedvisiontec.com`, 2008)
[2] An IEEE1394b bus with 800MBit$/s$ bandwidth is used.

6.1.3 Range Sensors

The 3DMo provides three range sensors, the **Laser Range Scanner (LRS)**, the **Light-Stripe Profiler (LSP)**, and the **Stereo Camera Sensor (SCS)**. Each range sensor is implemented as an individual application on the **Sensor PC**. All three range sensors are triangulation-based, however, the sensors differ e.g. in working range, accuracy, and robustness. In the following, LRS and SCS are explained briefly, whereas the LSP is discussed in more detail, as it is used as the exemplary range sensor for manual scanning in this chapter.

The Laser Range Scanner (LRS) [HDH97] is a 1D distance measurement device. It apllies the principle of point-wise laser triangulation. An outgoing laser beam is reflected diffusely on the object surface. The reflected light is collected by a receiver lens which focuses the light onto a *position sensitive detector (PSD)*. Additionally, the intensity of the emitted laser beam is controlled by the amount of received light. Thus, the system adapts dynamically to different reflection characteristics of the surface and changes in environmental lighting, ensuring robust measurements. The entire measurement unit is integrated into a rotating head, so that a stripe of distance measurements is generated at every turn. The sensor has a working range of $60\,mm$ to $300\,mm$ and a sampling rate of 400 points per turn at a rotation speed of 25 Hz.

The Stereo Camera Sensor (SCS) is an implementation of the **semi-global matching (SGM)** method introduced by Hirschmüller [Hir06]. Both camera streams are used to generate a single range image. Generally, stereo reconstruction provides good results on textured surfaces, but generates poor results on untextured objects. This capability is complementary to the ability of active range sensors. The SCS uses the full resolution of the cameras, allowing for acquisition of large areas at once. Due to the focal length of the cameras, the maximal working range is $250\,mm$ to $2000\,mm$. The resolution of the sensor is the camera resolution of 780×580 pixel. The SGM implementation has a measurement rate of 1 Hz.

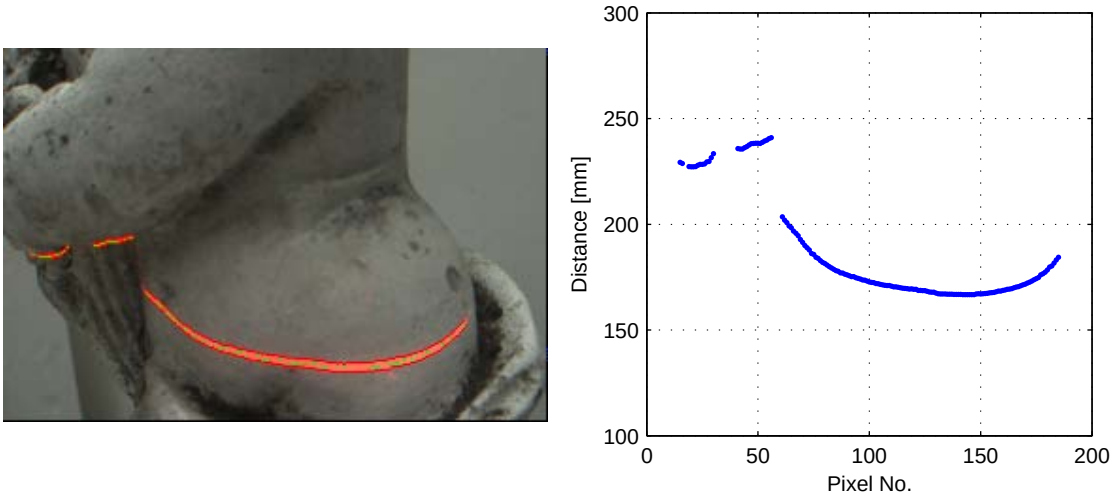
The Laser Stripe Profiler (LSP) [SSW+04] is a 1D range sensor that uses one of the cameras and the laser-line module. The laser beam illuminates a stripe on the surface while the camera records the diffuse reflection. An advantage of this system is that no optical filter[3] is used for simplification of the line segmentation in the image, as shown in Fig. 6.2. Thus, concurrent applications can simultaneously use the camera images. Possible applications are the SCS sensor, an image based pose estimation, or the use of the camera live stream in the visualization, as presented in Section 6.2.

The system provides distance measurements ranging from $100\,mm$ to $500\,mm$ at a resolution of 400 pixel and at a frequency of 25 Hz. In comparison to the LRS, the LSP can acquire larger surface regions at once, as it features a higher resolution and a larger working range. However, the system is less robust w.r.t. changes in surface properties and environmental lighting. In Tab. 6.1, the geometric parameters are summarized w.r.t. the geometric description in Chapter 2.2.

[3] Optical filters are mounted in front of the camera lens to filter light with a certain frequency, i.e. only the laser light can pass through.

Table 6.1: Parameters of the Laser-Stripe Profiler (LSP): The sensor features a perspective image coordinate system as described in Chapter 2.2.

N_u, N_v	u_0, v_0	$\Delta u, \Delta v$	$d_{\min}, d_{\max}$
$220, 1$	$-0.354848\,mm^{-2}, 0$	$0.00339948\,mm^{-2}, 0$	$100\,mm, 500\,mm$



(a) Segmented stripe (b) Distance profile

Figure 6.2: Function of the 3DMo-LSP sensor: Segmentation of the laser line (left) and distance profile (right).

6.1.4 Pose Measurement and System Calibration

The 3DMo system has couplers at the sides and at the rear to connect extensions to the system, as shown in Fig. 6.1. The couplers can be used either to attach the system to an external device (e.g. a robot), or to attach additional components needed for the calculation of the device pose (e.g. optical markers or an IMU[4]). This way, the system can easily be used in different configurations with various devices, e.g. mounted onto a robot or hand-held with optical tracking.

In this work, the system is configured as a hand-held device with an optical tracking system as pose sensor. The used pose sensor is a **smARTtrack IR-optical tracking system** from the Advanced Realtime Tracking GmbH[5]. The tracking system uses a calibrated set of infrared retro-reflecting markers to track the 6-DoF pose of the device. Thus, a unique and asymmetric marker configuration is attached to the lateral couplers of the 3DMo system. This hand-held setup is shown in Fig. 6.1(b).

The *smARTtrack* system supports synchronized measurements by an external hardware trigger. A hardware pulse generator is used for synchronizing the measurement times of the tracking system and the 3DMo components. Consequently, the *nearest pose* interpolation is used for this configuration (see Chapter 2.3).

The calibration for the *3DMo-LSP* sensor in combination with the tracking system consists of two major steps: first, the intrinsic and extrinsic camera parameters are determined w.r.t. to the coordinate system of the tracking markers. Secondly, the laser plane is calibrated relative to the camera coordinate system.

The intrinsic camera calibration and the extrinsic *eye-in-hand calibration* is performed

[4]IMU : Inertial Measurement Unit

[5]See `http://www.ar-tracking.com`, 2009

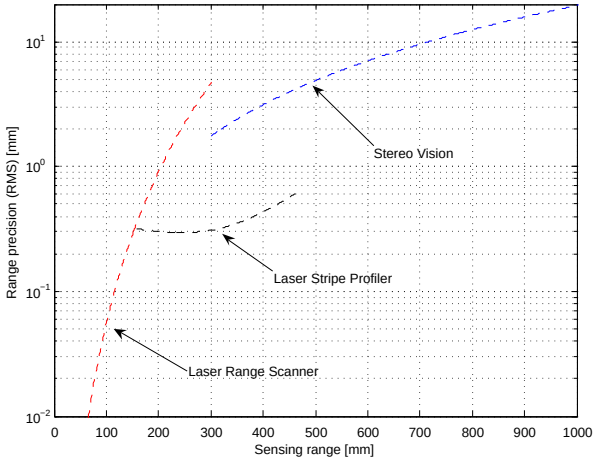


Figure 6.3: Precisions of the 3DMo range sensors: The plot shows the root mean square (RMS), i.e. the remaining error, after calibration. The precision of the LRS is very high in the close-up range, but decreases rapidly with an increasing distance. The LSP has a lower accuracy for short distances compared to the LRS, but its working range is broader and the precision does not decrease as rapidly. The SCS has the broadest working range, yet the lowest precision.

with the *CalLab tool box*[6]. A set of images from a chessboard-like 2D calibration panel and the corresponding device poses are captured. The landmarks in each image are detected fully automatically. They are used for estimating the intrinsic camera parameters including radial distortion and the extrinsic calibration. The laser plane is determined by sampling range images on a reference plane. The entire calibration procedure is described in detail by Strobl et al. [SSW+04] and Strobl and Hirzinger [SH06]. In Fig. 6.3, the precisions of the three range sensors of the 3DMo system are compared. The *LRS* system has a high precision in the close-up range but decreases rapidly with increasing distance. This is caused by the small base distance between laser source and optical sensor. The *LSP* has usually an lower accuracy for short distances compared to the *LRS* but its working range is broader and the precisions does not decrease as rapidly as for the *LRS*. The *SCS* has the broadest working range but also the lowest precision.
The expected deviation $\sigma(\tilde{d})$ can be determined by sweeps over a reference plane, as explained in Chapter 2.4. For the *3DMo-LSP*, a reference plane is used for the system calibration already. Hence, the deviation function can directly be estimated using the data from the calibration. For the *3DMo-LSP* in combination with the *smARTtrack* system, this experimentally estimated function is

$$\sigma(\tilde{d}) = -0.35367 + 0.0012484\,\tilde{d} + 1.026e^{-5}\,\tilde{d}^2 + 1.2655e^{-10}\,\tilde{d}^4 \tag{6.1}$$

and is further used for calculation of the per-point expected deviation, as used in the *RT-SSR* method.

6.2 Visual Feedback for Manual Scanning

As stated in Chapter 1, visual feedback is mandatory for manual scanning. A human operator requires a visual feedback in order to monitor the sampling progress. This **human-in-the-loop** system has been illustrated in Fig. 1.1 on page 2. Therefore, the 3D model that is generated by the *RT-SSR* is connected to a real time visualization.

[6]The DLR Camera Calibration Toolbox (CalLab) `www.robotic.dlr.de/callab`, 2009

Table 6.2: Parameters of the RT-SSR for the DLR-LSP

Parameter	**Symbol**	**Value**
Limitation radius	$R_{\min}$	$0.5\,mm$
Override for initial neighborhood	$R_{n_{user}}$	$3\,mm$
Max. number of neighbors	k_n	20
Max. valid grazing angle	$\alpha_{s_{\max}}$	$75°$
Max. fast selection angle	$\alpha_{n_{\max}}$	$5°$
Min. number of selected neighbors	$n_{s_{\min}}$	5
Min. edge length	$R_{e_{\min}}$	$0.5 - 2\,mm$
Max. edge length	$R_{e_{\max}}$	$3\,R_{e_{\min}}$

In Chapter 5, the parameters of the *RT-SSR* method have been analyzed. In the following section, the settings for the *3DMo-LSP* are summarized. Further, the design of a visualization for visual feedback is discussed w.r.t. the rendering of the geometry, the adjustment of the virtual camera, and an optional augmentation of the visualization.

6.2.1 Parameterization of the RT-SSR processing

As analyzed in Chapter 5, the *RT-SSR* method must be adapted to the scanner system used, as an unsuitable parameter set may cause poor modeling results. For the *3DMo-LSP* sensor, the settings in Tab. 6.2 are applied. The initial neighborhood radius R_{n0} is determined dynamically by Equation (5.1). Hence, only the manual override $R_{n_{user}}$ is set. It is used to define the maximum distance between two consecutive and parallel stripes on the surface, which enables the sample points of one stripe to be inside the respective neighborhood of the other stripe's points. The thresholds of the selection stage are set confirming the recommendations developed in Chapter 5. The minimum edge length $R_{e_{\max}}$ can be set application-dependent by the user.

6.2.2 Visualization

Different approaches to real time rendering or visualization of 3D data exist. They can be categorized as point rendering methods, surface rendering, and volumetric visualization. A summary of concepts for real time rendering is found e.g. in the textbook of Akenine-Moeller [AMH02].
A simple approach to providing visual feedback for manual scanning is to render the raw 3D points. This "brute force approach" has two major disadvantages: First, all data must be loaded into the GPU. This data overhead potentially results in a slow rendering of the point model. Secondly, a suitable shading[7] is not possible, as no information concerning the surface gradient is available. More advanced rendering methods typically require information concerning the local gradient of the surface. Even point rendering methods need at least the surface normal for every point, e.g. in the *QSplat* method of Rusinkiewic and Levoy [RL00]. However, in the context of visual feedback for manual scanning additional requirements exist. As the visualized 3D model is continuously changing, the visualization should not require a complex preprocessing of the data, which would have to be rerun every time the 3D model changes. An example for such an undesired preprocessing is the arrangement of the data in a level-of-detail graph that has to be completely rebuilt with every data change.

[7] Shading is the depicting of depth in 3D models by varying levels of darkness.

In this work, a classical rendering of triangles is used, as the streaming surface reconstruction generates triangle meshes and no further conversion of the data is required. Moreover, even older graphics hardware supports the acceleration of triangle sets. The triangles and edges of the mesh are loaded to the GPU and stored as a so-called display list. This display list is updated, every time the model changes. However, not every change of a vertex, edge, or triangle requires an update of the list. Typically, the model is updated after a complete range image has been processed. For single stripe systems with a high measurement frequency (e.g. the *3DMo-LSP*), an update of the visualized model is performed after the integration of $5 - 10$ new range images. This simple rendering concept is sufficient for smaller models, e.g. busts or technical parts, since recent graphics adapters can render models with more than $200,000$ triangles in real time. Other techniques, such as point rendering or level-of-detail extensions that use e.g. the octee structure are possible, but are not discussed in this thesis.

6.2.3 Virtual View and Augmentation

A suitable virtual view of the generated 3D model is essential for a good impression and overview during the scanning progress. For the inspection of the model during and after the scanning, an interactive view control is suitable, e.g. interactive zoom, translation, and rotation functions as used in numerous 3D viewer and CAD programs. However, during the scanning process, the view must be adapted dynamically w.r.t. the changes in the triangle mesh. Here, the following principal view adoption strategies are applicable:

- Fit of the virtual view to show the entire scene
- Fit of the virtual view to show the last changes in the mesh
- Alignment of the virtual view to the scanner pose

In this visual feedback application, the last mode, *alignment of the virtual view to the scanner pose*, is used during the scanning process and for the inspection task. The alignment allows a use of the scanner itself as an input device, i.e. the operator can change the view by moving the scanner and without the need to discard the device.
The visualization can be additionally improved by augmenting the displayed model with a live camera stream from the sensor system. Here, the camera images are rendered temporal before the generated mesh is displayed, i.e. the image is behind the model. The opening angle of the virtual view must be set to the values of the real camera, as otherwise, the object is scaled differently than the background live image. This *mixed reality* visualization, i.e. augmentation of the generated 3D model by the real scene, aids the operator in navigating the system across the object's surface and thus increases the usability of the system. Especially for hand-guided scanner systems that allow for a full 6 DoF movement, this feature is helpful.
The overall concept of the visual feedback system is illustrated in Fig. 6.4. The *RT-SSR* method is the core component of the system. It is fed by a stream of 3D points from the scanner system. The resulting triangle mesh is visualized in front of the live stream image. The raw stream of sensor poses is used in a virtual camera to calculate the correct view onto the model.

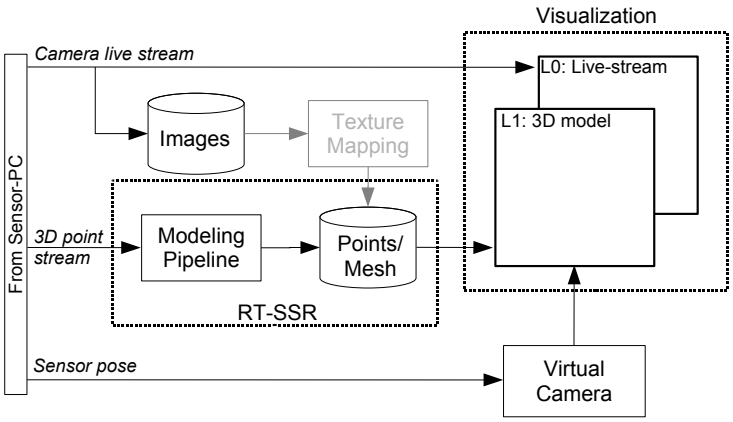


Figure 6.4: Concept of the 3DMo visual feedback system: The core component is the *RT-SSR* method. It is fed by a stream of 3D points from the scanner system and supports the visualization with the resulting triangle mesh (L1). The camera live stream is rendered behind the 3D model (L0). The sensor pose stream is used to align the virtual view onto the mesh with the real sensor pose. Further, images from the camera stream can be captured and used for texture mapping after acquisition.

6.2.4 Texture Mapping

Texture mapping adds a photorealistic impression to a given 3D model by linking each of its surface elements with a realistic image, called texture. Often, predefined synthetic textures are used for 3D models. In the context of manual scanning, real images of the object are available and can be used as texture of the model.
Beside the mesh generation and the instant visualization of the generation progress, the visual feedback system provides the capture and storage of single color images from the camera live stream, as illustrated in Fig. 6.4. The captured images can be mapped onto the generated surface in a post-processing step, i.e. this processing is performed after acquisition and surface reconstruction.
The texture mapping relates a part of an image part to each triangle in the generated mesh. Generally, it is performed by projecting the images onto the model. The mapping requires known intrinsic camera parameters including lens distortion and camera pose relative to the 3D model. For the 3DMo system, the intrinsic parameters are determined by camera calibration as described in Section 6.1.4, and the camera pose is measured by the pose sensor. However, a single triangle face of the mesh can be seen from multiple camera views, potentially causing mapping ambiguities. Three methods are developed to handle these ambiguities and summarized in the following. A detailed description is given by Hirzinger et.al. [HBH+05]:

- **Single-View Mapping:** Each triangle with a surface normal that points sufficiently in view direction is mapped to camera coordinates, using the camera parameters and pose. The degree of detail (i.e. the area of the projected triangle) is computed and the view with the highest degree of detail is associated to the examined triangle face.

- **Single-View Mapping with Brightness Correction:** Again, a single view is associated to a triangle face. Due to non-lambertian reflectance characteristics of the surface, the perceived brightness of the face can vary from view to view. To compensate these variations, contrast and brightness of each camera image are adjusted.

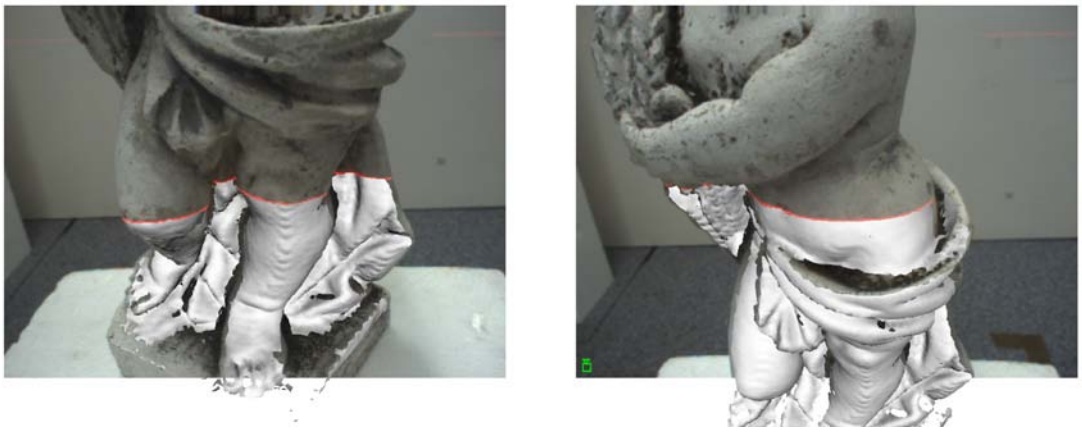


Figure 6.5: Mesh generation and augmented visualization of the progress at the digitization of a statue: A sweep over a cherub statue is instantly processed and the resulting mesh is visualized. The 3D model (white shaded) is rendered in front of the camera live stream with a virtual view that is aligned to the real scanner pose. This visual feedback helps the user to navigate the scanner devise w.r.t. the statue.

- **Multi-View Mapping:** A single triangle face is mapped to all camera images, if it points sufficiently into the view direction of the respective image. The texture image for the triangle face is computed by a weighted averaging over the images. The weights represent the degree of detail of the projected triangle in the respective image.

6.3 Results

The *3DMo* with the visual feedback system is suitable for numerous applications. In the following, results of two exemplary application fields are depicted: the generation of photorealistic models and the registration of objects.

6.3.1 Generation of Photorealistic Models

The *3DMo* in the hand-held configuration is used for the 3D acquisition of smaller objects, such as busts or other objects for archiving of cultural heritage and the capture of technical objects for rapid prototyping. An application is the generation of photorealistic 3D models e.g. for the use in virtual reality applications, for documentation, or for rapid prototyping.
Fig. 6.5 shows the process of mesh generation and visualization with augmented camera live stream for the example of a putto statue scanned with the *3DMo-LSP*. A mesh resolution of $R_{e_{\min}} = 1.5\,mm$ is used. The 3D model is rendered in front of the camera live stream with a virtual view that is aligned to the real scanner pose, as described in the preceding section. In Fig. 6.6, the resulting mesh and the textured model are shown.
In Fig. 6.7, two further results of the *RT-SSR* method are depicted. Both meshes are generated with a resolution of $R_{e_{\min}} = 0.5\,mm$. For the Zeus bust (a,b), two sweeps are performed. The computing time is $5.03\,s$ for $65,381$ input points. For the Mozart bust (c,d), three strongly overlapping scans are acquired. The computing time is $6.84\,s$ for $106,929$ input points. This example demonstrates the efficiency of the method w.r.t. to

(a) Generated mesh (b) Textured model

Figure 6.6: Generated 3D model of a putto statue: The generated mesh after two sweeps consists of $8,6162$ vertices and $166,475$ triangles (a). Further, the a captured camera image is used to map a texture onto the model (b).

redundant input data and approves that the processing time is barley coupled to the size but only to the resolution of the mesh.
In Fig. 6.8, the textured 3D models of more objects for different applications that have been digitized with the 3DMo system are illustrated. The Indian bust can be used in virtual catalogs or as object in VR-scenes. The box of pipes is digitized and used for grasp planning. The meteorite model is generated from a part of the *Neuschwanstein-meteorite* that dropped to earth in 2002. This model was generated for evaluating the documentation with 3D models.

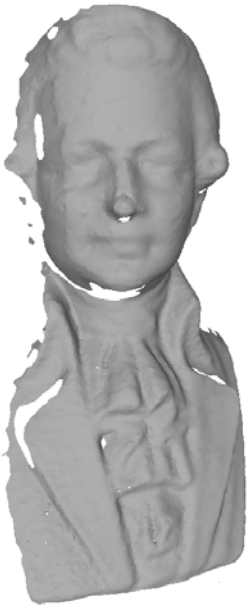
6.3.2 Object Registration

Another application for manual scanning and for the 3DMo system is the **object registration**. The goal of object registration is the alignment of a 3D model with another *template* 3D model by estimating the rigid body transform between both. A typical application for registration is to fit an artificial 3D model, e.g. a CAD model, to a real object. However, for a robust registration, it is required to sample the object sufficiently and to include characteristic parts, e.g. edges and concavities that are specific to the object. The use of a manual scanning system is beneficial to enhance the handling of the registration setup, as the visual feedback system enables the operator to inspect the scanned model before registration as well as the registration result. If the registration fails, e.g. because the object is not sufficiently scanned, it is possible to add more scans to the model and repeat the registration. This interactive scan-and-register approach enables even non-experts to handle such a system.
Registration can be divided into local optimization methods, e.g. the *Iterative Closest Point (ICP)* algorithm by Besl and McKay [BM92] or one of its numerous variants, and global methods, e.g. the methods of Barequet and Sharir [BS97] or of Gelfand et al. [GMGP05]. The key challenge of object registration is the correct matching of corresponding points in both models. The ICP method uses a nearest neighbor strategy in order to identify corresponding points and thus requires a good initial transform. Global registration methods typically use geometrical features for finding corresponding points. Hence, a surface model or at least the surface normals for each point are required in both models.

(a) Scans

(b) Mesh

(c) Scans



(d) Mesh

Figure 6.7: Scans and 3D models of two busts: The Zeus (a) and Mozart (b) busts are scanned using the 3DMo-LSP. For Zeus, two sweeps are performed (a), and for the Mozart three strongly overlapping sweeps are acquired (c). The corresponding meshes (b,d) are generated with a resolution of $R_{e_{\min}} = 0.5\,mm$. The 3D model of the Zeus bust consists of $61,556$ triangles and $31,733$ vertices. The Mozart model contains $67,301$ triangles and $34,120$ vertices.

(a) Indian bust (b) Box of tubes (c) Meteorite

Figure 6.8: Photorealistic 3D models for miscellaneous applications: An Indian bust for virtual catalogs (a), a box of pipes used for grasp planning (b), and a meteorite for documentation purposes.

The computational effort of object registration methods increases with the size of the model and its resolution. For the ICP method, the correspondences for each point of the scanned model have to be found. Here, it is sufficient to represent both models as a point set. The limitation stage of the *RT-SSR* method helps reducing redundancies in the scanned model and thus avoid a high computational effort. Further, the estimated surface normals can be applied to increase the robustness of the registration. For feature-based registration methods, the mesh is used to derive curvature-based features. The *RT-SSR* method reduces the effort for feature calculation and thus for registration, as the mesh resolution can be chosen task-dependent and thus the size and resolution of the generated model is sufficiently scaled.
The registration with manual scanning can be used for various applications. In the following, two exemplary applications are introduced: the patient registration for robot-assisted head surgery and the work piece registration for inspection and small batch production.

Robot-assisted head surgery In robot-assisted head surgery, the intervention is typically planned before, in a pre-operative phase. Here, MRI or CT data is used to plan trajectories. The registration is required to transfer the planned intervention to the pose of the patient in the intra-operative phase. The traditional method for registration is to implant markers in the skull that are visible in the CT, to measure the marker during the intervention with the robot, and to match these with the CT model. Alternatively, the face of the patient can be scanned and this model can be matched with the CT or MRI model. This method is non-invasive and thus more gentle for the patient. This was evaluated with the *3DMo* system by Korb et al. [KBE+04] and Konietschke et al. [KBB+07]. Fig. 6.9 shows an exemplary result of a scanned face that is matched with an face model from MRI.

Work piece registration Small batch productions are typically not automated, as a full automation is mostly not profitable. However, it is desired to automate certain production steps and to help the worker with interactive tools. Here, registration can be used to match work pieces on a workbench with a CAD model either for inspection of the production or for transferring automatic and semi-automatic planning.

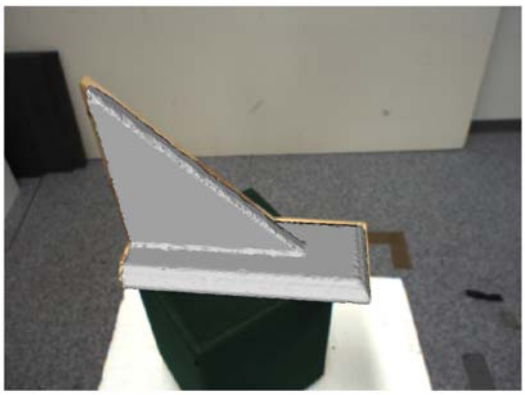
(a) Operating room setup (b) Registration result

Figure 6.9: Registration of scanned faces (white) with MRI data (red)

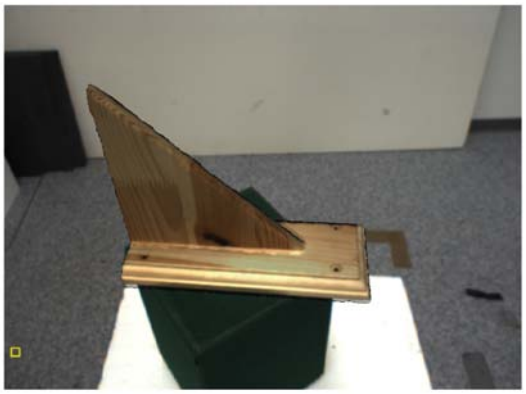
Fig. 6.10 shows the example of a wooden work piece that is partially scanned and then registered with a synthetic template model, i.e. the template is fit to the scanned data. Hence, a worker can visually inspect the object. Further, real texture is mapped to the template object. For woodworking the possibility of generating textured objects is beneficial, as the grain of a wooden piece is potentially important for further processing.

In summary, the use of the *RT-SSR* method enables for an enhanced visual feedback in the context of manual scanning. The shown exemplary applications demonstrate the versatile usability of manual scanner systems, if a suitable visual feedback system is provided. However, in the next chapter another application to the *RT-SSR* method beyond manual scanning is shown.

(a) Scanned model



(b) Template model



(c) Textured template model

Figure 6.10: Inspection of the registration for a wooden workpiece: The wooden work piece is scanned (a) and the synthetic template object is fitted to the scan data (b) and can be visually inspected. After registration, a real texture is mapped onto the template object (c).

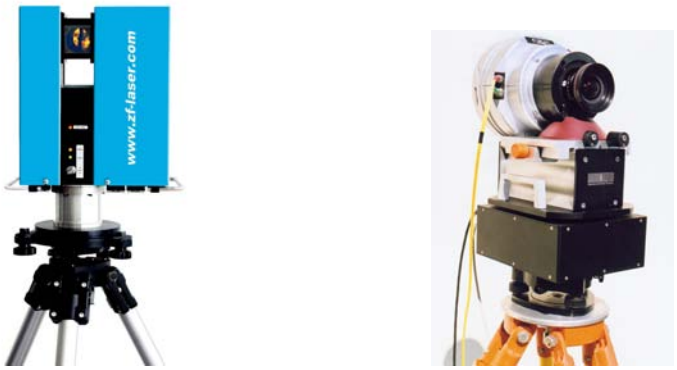
7
Large Object Modeling

This work focuses on streaming mesh generation from real time 3D measurements to be used as advanced visual feedback in manual scanning applications. However, applications beyond this primary purpose are possible. In this chapter, the application of the *RT-SSR* method for generating surface models from huge data sets, i.e. sets containing millions of points, is introduced. In this context, only the point-by-point integration of the data is important, not the instant availability of the model, as the data is processed after acquisition. Further, a visual feedback during the acquisition of huge 3D models is not possible with most of today's commercial 3D rendering tools and hardware due the combination of data size and dynamic behaviour.

The presented modeling of large objects is part of a project aiming at increasing the automation of model generation for 3D documentation and archiving of buildings using laser scanners and color cameras. Typical applications the preserving of cultural heritage, e.g. castles or churches, or the 3D documentation of the current state of industrial buildings, e.g. factories. The project is a cooperation between the departments *Robotic Systems* and *Optical Information Systems* of the *DLR Institute of Robotics and Mechatronics* as well as the companies *Zoller and Fröhlich GmbH* and *Illustrated Architecture*. First results were published by Hirzinger et al. [HBH+05] and Liu et al. [LSHB07].

Buildings are typically documented in constructional drawings, i.e. 2D maps. In addition, so called ortho-images of walls and ceilings are generated. An ortho-image is the orthographic projection of images onto a projection plane. This technique is limited to simple geometries, since occlusions and non-planar objects cause distortions in the image. The generation of 3D models is an alternative which overcomes the limitations of ortho-images and thus improves the documentation.

Today's long-range laser radars enable a fast acquisition even of large geometries and thus the fast digitization of rooms and buildings. However, the generated range images are huge and have large overlapping areas. The data is hardly usable, as commercial software suites typically fail to handle this amount of data. A surface model decreases the redundancy in the data and thus the overall size of the model. Hence, the the required memory consumption in subsequent processing steps is reduced significantly, and it is easier for visualization tasks and other applications to handle the 3D model. Further, surface models enable the generation of photorealistic 3D models



(a) Z+F Imager 5003 (b) *Panoramic Camera*

Figure 7.1: The *Zoller + Fröhlich Imager 5003* (Source: `http://www.zf-laser.com`, 2009) and the *EyeScan M3metric* Panoramic Camera developed at DLR, Berlin in cooperation with KST GmbH, Dresden (Source: `http://www.kst-dresden.de`, 2009).

by mapping color information to the surfaces. Hence, an advanced 3D documentation of buildings and further applications, e.g. virtual walk-throughs or reconstruction of missing parts of a building, are possible.
Several methods for streaming surface reconstruction for such out-of-core applications have been published, e.g. the methods of Pajarola [Paj05] and Allègre et al. [ACA07]. Although the *RT-SSR* method is not primarily designed for this application, it is suitable for out-of-core processing. An advantage of this method is the implicit sorting of the data, i.e. the data need not to be spatially sorted before processing.
In the following, the application for the *RT-SSR* method to out-of-core processing of huge data sets is shown. First, the used hardware and necessary preprocessing steps are introduced. Afterwards, the adaption of the *RT-SSR* method, its parametrization, and further post-processing steps are explained. Finally, exemplary results are shown.

7.1 Data Acquisition and Preprocessing

The on-site work and the preprocessing of the scan data have mainly been performed by *Illustrated Architecture* and the *DLR* department *Optical Information Systems*. In the following, the scanner system and necessary processing steps are summarized.

7.1.1 Scanner System and Color Sensor

The acquisition of the geometry is performed with a long-range laser radar and hi-resolution color information is added by a panoramic camera. In the following, both sensors are briefly introduced.

Scanner System The long-range laser radar *Z+F Imager 5003*[1] is used for acquiring the geometry. The system is shown in Fig. 7.1(a). The optical unit of the system sends an IR laser beam and uses the time-of-flight principle to calculate a distance from the reflection. It is actuated by two rotating axes. They compose a spherical coordinate system with a (v, u)-order, contrary to the (u, v)-order spherical system descrbed in

[1] Zoller + Fröhlich Imager 5003 : See `http://www.zf-laser.com`, 2009 for details.

Chapter 2. Hence, the mapping of a distance pixel d at the grid coordinates u, v in the spherical image coordinate system to Cartesian coordinates is

$$\mathbf{p}(u,v,d) := \begin{pmatrix} d \cdot \cos v \cos u \\ d \cdot \cos v \sin u \\ d \cdot \sin v \end{pmatrix} ,$$

and the corresponding line-of-sight is

$$\mathbf{s}(u,v) = \begin{pmatrix} \cos v \cos u \\ \cos v \sin u \\ \sin v \end{pmatrix} .$$

further, the reference sample density is

$$\delta(d,u,v) = \min\{2\,d\,|\sin v(j) \sin\frac{\Delta u}{2}|)^{-1}, (2\,d\,|\sin\frac{\Delta v}{2}|)^{-1}\} .$$

The system has a working range of $1\ m$ to $53\ m$. It can be configured for different resolutions with up to $20,000$ pixels per axis. In this work, a resolution of $10,000 \times 5,000$ pixels is used, i.e. each range image consists of 50 million pixels. The measurement error of the system is modeled by a Gaussian zero-mean distribution with a distance-dependent deviation[2]

$$\sigma(\tilde{d}) = 0.375 + 0.0675\,d + 0.0015\,d^2 .$$

Here, the error of the pose measurement need not to be considered, as the pose is estimated and optimized after acquisition, as described in the next section.

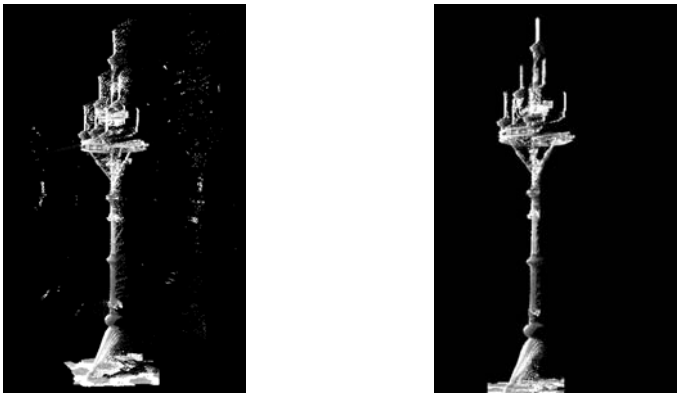
Color Sensor The *EyeScan M3metric Panoramic Camera* is used to gather high-resolution color information that can be used for texture mapping. It was developed in a common project between DLR and the *KST GmbH*[3].
The camera integrates three $10,200$ pixels *line CCD chips*, one for each (RGB-) color channel and uses a special optical lens system of $35\ mm$ focal length. It is mounted onto a rotating unit which contains an angle-increment measuring system with a resolution of $0.001°$. Hence, color images with resolutions of up to $10,200$ by $360,000$ pixels are possible. A typical image acquisition with $10,200 \times 30,000$ pixels takes about three minutes at daylight and up to 60 minutes at a dark indoor illumination. The time mainly depends on the ambient illumination conditions and the number of rows to be measured with the camera. Detailed information on the technical specification and is provided by Scheibe et al. [SKR+01], the camera calibration and applications are summarized in [Sch06]. The application of the system to 3D modeling is described by Hirzinger et al. [HBH+05].

7.1.2 Preprocessing of Range Images

After acquisition, the range images are filtered and the scan pose is calculated. The filtering removes outliers and other artifacts in the range images that would otherwise affect the pose calculation and surface reconstruction. The pose is estimated from the scans, as this setup provides no sensor to measure the pose at acquisition time. This filtering and pose calculation are not part of this work, but are summarized in the following for a better understanding.

[2]The deviation has been extracted from the noise details in the data specification of the *Z+F Imager 5003*. `http://www.zf-laser.com`,2009

[3]*KST GmbH, Dresden*, `http://www.kst-dresden.de`, 2009)



(a) before filtering (b) after filtering

Figure 7.2: Automatic filtering of range images: The filtering of range images is shown for the measurement of a shiny golden candlestick. The range image contains outliers and a tail of very noisy samples behind the object caused by specular reflections (a). The automatic processing with the filters eliminates most of the undesired points (b). Source: [Sch06]

Scan Filtering

Range images may contain invalid data, e.g. outliers or other artifacts, that typically result from shiny or transparent objects. The *RT-SSR* method is principally able to filter outliers implicitly, however, an range image preprocessing that removes undesired artifacts enables a more accurate mesh generation, as otherwise these mismeasurements must be considered in the parameter settings. Scheibe [Sch06] describes a set of median filters, histogram filters, and spike detectors that are used for filtering. The filtering is performed fully automatically for each range image. It allows for a robust detection of outliers and other erroneous distance measurements, even for shiny objects. Fig. 7.2(a) shows errors occurring during the measurement of a shiny golden candlestick, Fig. 7.2(b) illustrates the corresponding filtered range image.

Scan Pose Calculation

Before surface reconstruction the images have to be aligned correctly. Here, both, automatic and manual approaches were tested. The reference method is the photogrammetric alignment with artificial markers. Here, the Neptan software suite[4] is used for alignment. Alternatively, automatic registration algorithm of Liu and Hirzinger [LH05] can be used. This method allows for combining multiple 2.5D scans without the need of placing wall markings or performing additional theodolite measurements. Instead, the overlapping areas between the scans are used to estimate how they are related. This marker-free approach is particularly suitable for historic buildings and other ancient sites, as it is often prohibited to install wall markings. The registration process is divided into three steps: a segmentation of each range image into patches with homogeneous curvature, a coarse matching of the data with a matching-tree algorithm, and a fine matching, optimizing the alignment by a set of control-point-pairs. A detailed description of the method is given by Liu and Hirzinger [LH05].

[4] See `http://www.technet-gmbh.com` for details.

Table 7.1: Parameters of the RT-SSR for the *Z+F Imager 5003*

Parameter	**Symbol**	**Value**
Limitation radius	$R_{\min}$	$4\,mm$
Override for initial neighborhood	$R_{n_{user}}$	$15\,mm$
Max. number of neighbors	k_n	20
Max. valid grazing angle	$\alpha_{s_{\max}}$	$75°$
Max. fast selection angle	$\alpha_{n_{\max}}$	$5°$
Min. number of selected neighbors	$n_{s_{\min}}$	5
Min. edge length	$R_{e_{\min}}$	$> 4\,mm$
Max. edge length	$R_{e_{\max}}$	$3\,R_{e_{\min}}$

7.2 Surface Reconstruction and Post-Processing

After preprocessing, the range images are transformed into sets of 3D points with their corresponding attributes. The *RT-SSR* method is used to generate an initial mesh from the point sets. Accordingly, potential holes in the model are filled, the mesh is simplified, and texture is mapped onto the generated surface.

7.2.1 Surface Reconstruction

The points are processed with the *RT-SSR* method by successively inserting them (streaming). For this application, the two principle stages of the processing pipeline, as illustrated in Fig. 3.1, are performed individually, i.e. the surface normals for all points are estimated before the mesh generation is performed. Tab. 7.1 shows the selected parameters for the *RT-SSR* method.

The *RT-SSR* method does not require to load the entire input data at once into memory but processes the data point-by-point, similar to other out-of-core methods. However, the method must store the limited sample point set and the resulting mesh in the memory. Hence, the size of the mesh, i.e. its extension and resolution, that can be processed at once is limited by the physical memory of the used computer system. The used octree data structure can principally be extended by a smart swapping strategy that overcomes this limitation. However, this extension is beyond the scope of this thesis. As a consequence, huge 3D models with high resolution can not be processed at once but must be divided into smaller parts.

In this work, the acquired space, e.g. a single room, is divided into a set of overlapping boxes, similar to the voxelization presented in Chapter 4 and each box is processed separately. Only the 3D points that are inside the respective box are passed to the *RT-SSR* steps. The overlap of the boxes is necessary to avoid incomplete a surface reconstruction in the border area of the boxes, caused by sparse or unbalanced neighborhoods. The box generation can be performed automatically by defining the number of boxes for each direction and the percentage of overlap. The bounding box of the combined set of all 3D points is calculated and segmented into the desired amount of boxes. However, if already some construction plan or other information concerning the scanned scene exist, a manual definition of the boxes is superior to the automatic approach, as walls and other separating elements can be taken into account and avoid that e.g. the boarder of a box is a wall or single boxes contain only a very small part of an object.

7.2.2 Post-Processing

The surface reconstruction generates a dense and homogeneous mesh for each processed box. These meshes are further refined and merged into a single model in a set of automatic post-processing steps. These steps are described in detail by Hirzinger et al. [HBH+05] and Liu et al. [LSHB07] and are summarized in the following:

Hole filling and clean-up The mesh may contain holes that originate from areas not sampled e.g. due to occlusions or transparent materials. Further, artifacts that are not filtered in the preprocessing may result in small islands, single triangles, or other artifacts in the mesh. These artifacts are removed and smaller holes are filled. However, large holes, e.g. windows, are not closed, as this might result in larger differences between the model and the real geometry.

Simplification The 3D models of building interiors typically contain a large amount of plane areas, e.g. walls. The resolution of the mesh in these areas is reduced, thus the model is simplified. A simple method is used to perform an initial simplification: The curvature is analyzed by comparing the surface normals of adjacent triangles. If the curvature in an examined area is below a certain threshold, the mesh is locally re-triangulated with a lower resolution. For 3D models of rooms and halls, this simplification typically removes about 90 percent of the triangles.

Smoothing The simplified model is smoothed in order to reduce potential sensor noise. A fairing algorithm presented by Liepa [Lie03] is used, as this method provides a smoothing with less global shape distortion compared to other methods.

Merging The separately processed boxes or parts can be merged after simplification and smoothing. This step is optional and application-dependent, as many commercial programs are not able to handle single models of arbitrary sizess.

Texturing One goal of the automatic processing is the generation of photorealistic models. Hence, the acquired color images are mapped onto the refined geometry. The mapping of panoramic camera images to the generated surface is part of the work of Scheibe [Sch06]. Here, a ray casting method considering the specific camera geometry is introduced.

7.3 Results

The presented automatic processing has already been for numerous projects. In the following, exemplary results for the 3D documentation of *Neuschwanstein Castle* are shown and further applications are demonstrated.

7.3.1 Documentation of Castle Neuschwanstein

Neuschwanstein Castle was digitized in order to create a complete set of architectural drawings and ortho-images. Here, the Z+F Imager was used to digitize all rooms of the castle, as it rapidly digitizes building interiors i.e. rooms, halls and facades, thanks to its large working range. However, multiple scans from different views are required due to occlusion by e.g. pillars, handrails, furniture, etc. 450 rooms, staircases, and

corridors were acquired, resulting in a total amount of over 2000 scans with about 50 million points each. The scans were aligned by fiducial marks that were attached to the walls during acquisition. Further, color images of multiple rooms were captured with the *EyeScan M3metric Panoramic Camera*.
This data is used to evaluate the automatic model generation described in the preceding section, for an enhanced documentation of the castle. Fig. 7.3 shows an overview of all modeled parts of the castle. For the generation of an initial mesh, the parameters listed in Tab. 7.1 and a mesh resolution between $R_{e_{\min}} = 4\,mm$ and $R_{e_{\min}} = 8\,mm$ are used.
Fig. 7.4 shows the king's room in the gatehouse. The processing of this room requires no partitioning into boxes, as it is relatively small. After post-processing, the model contains of $410,309$ vertices and $770,422$ triangles.
A photorealistic model can be generated either using the intensity images of the *Z+F Imager*, as shown in Fig. 7.5(a), or applying color images from the *EyeScan M3metric Panoramic Camera* shown in Fig. 7.5(b). As an furhter example, Fig. 7.6 shows the color textured model of the famous throne room of the castle.

Figure 7.3: Modeled parts of Neuschwanstein Castle: 450 rooms, staircases, and corridors were acquired and a 3D model was generated for each, using the automatic processing steps described in this chapter. A maximum mesh resolution between $4\,mm$ and $8\,mm$ is used.

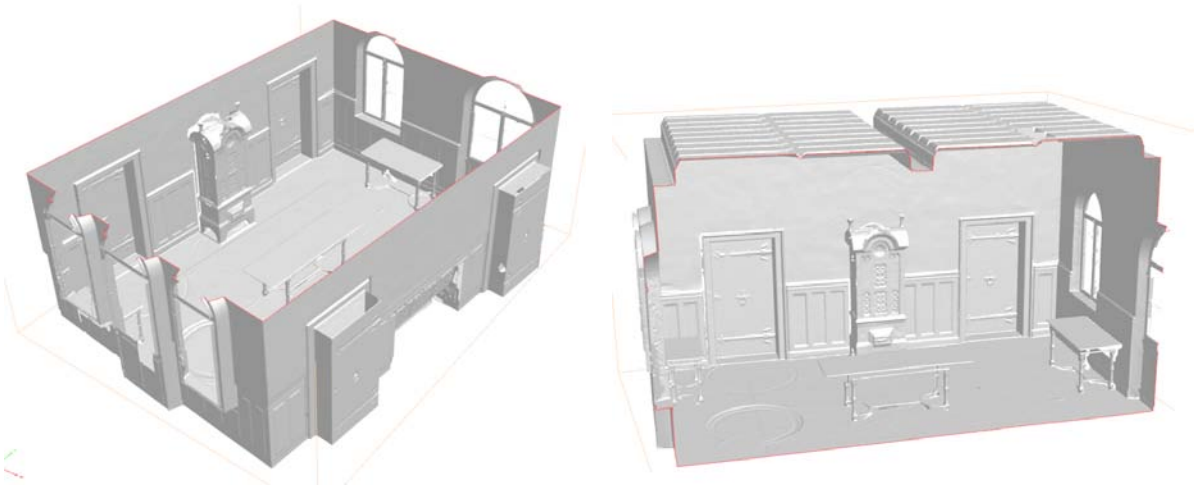
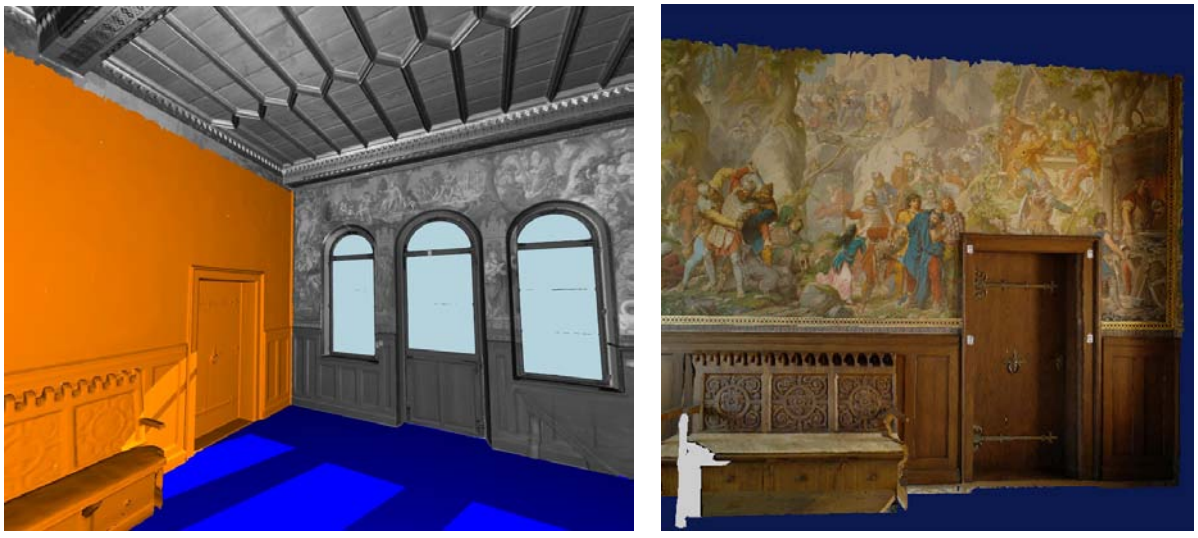


Figure 7.4: Model of the king's room in the gatehouse: After post-processing contains $410,309$ vertices and $770,422$ triangles.



(a) Intensity texture (b) Color texture

Figure 7.5: Texturing of the models: Either intensity images of the Z+F Imager are used (a) or color images from the DLR panoramic camera are applied (b).

Figure 7.6: Throne room of Neuschwanstein Castle with applied texture. Source: [Sch06]

7.3.2 Further Applications

Another application example for the presented automatic modeling is the reconstruction of the roof of a historical house. The existing parts are scanned and a surface model is generated, as shown in Fig. 7.7. The latter can be used by architects in a CAD software suite to reconstruct missing parts.
A technical application of the modeling of building interiors is the documentation of the current state of industrial buildings, e.g. a factory. A major problem of planning alterations of aged industrial buildings is that the real state of the building differs from the initial construction plans. As an example, pipes or machines may have been installed or changed without documentation. Hence, the alteration of parts of the building or the installation of new machines becomes problematic or even impossible, as undocumented pipes or machines block the required space. The rapid documentation of the current state with an 3D model supports designers or architects in the planning stage of new installations or alterations. Moreover, differences to former states can be calculated and thus, temporal changes can be tracked. In Fig. 7.8, a part of an factory building surface model is shown and compared to the raw point data.

These examples show the suitability of the *RT-SSR* method for the processing of huge data sets, even though it is primarily designed for visual feedback applications in the context of manual scanning. The introduced processing steps increase the automation of the overall processing of these data sets and requires only selected user interactions. This is an important step towards processing tools that are usable for everyone. The resulting 3D models decrease the amount of data significantly and support further processing with commercial software suites.

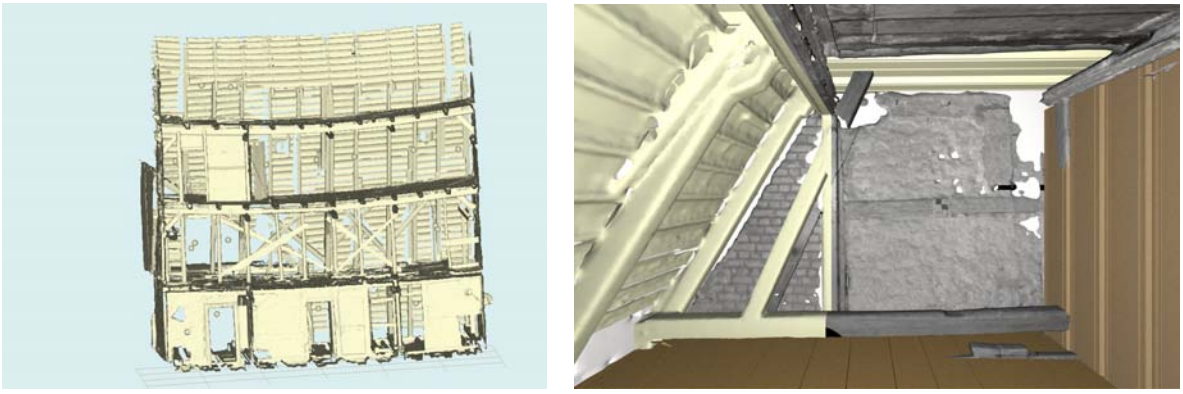


Figure 7.7: Reconstruction of a roof: Overview of the scanned and modeled parts (left) and the reconstruction of walls and floors in the model (right).

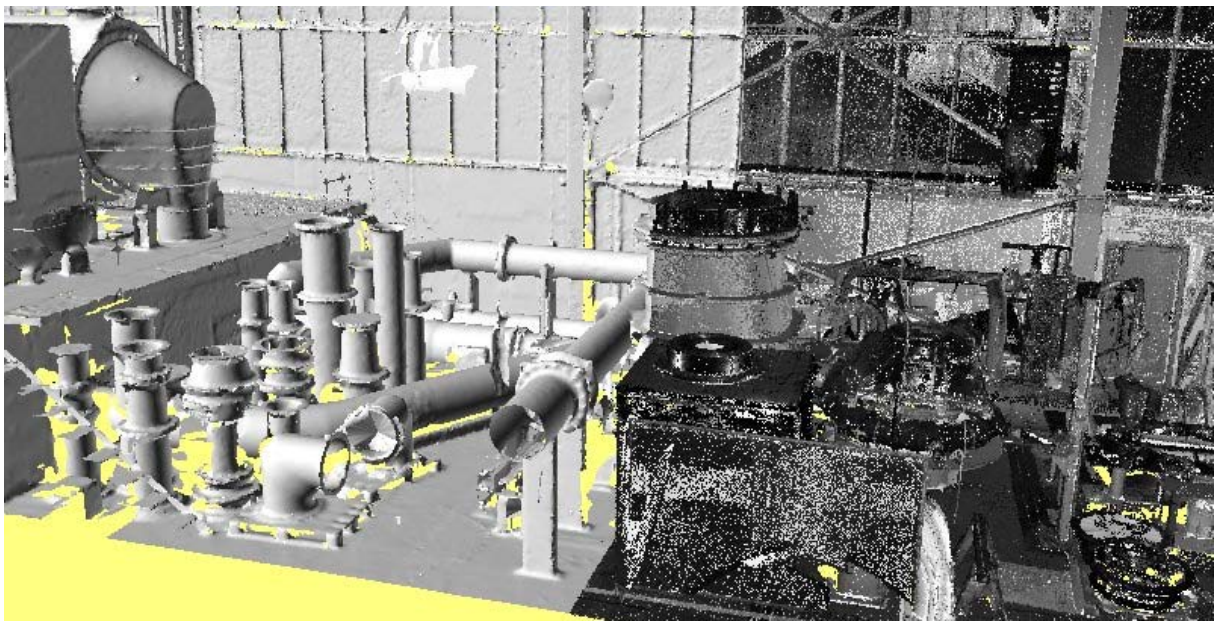


Figure 7.8: Model of a factory building: The generated surface model (left) and the raw point data (right).

8 Conclusion

This chapter provides conclusive remarks on the achievements of this thesis as well as an outlook to potential future applications and future research.

8.1 Conclusion

In this thesis, an approach for streaming surface reconstruction for real time 3D measurements in the context of visual feedback for manual scanning is introduced. The method generates a triangle mesh directly from the data stream of a scanner system, i.e. a 3D model is created incrementally during the acquisition. This approach differs from typical streaming methods that usally focus on out-of-core applications, since the real time character of the problem implicates an unknown object size and number of sample points.
This method is suitable for unorganized point sets and is not limited to a certain type of scanner, as the measurements enter the surface reconstruction process as a stream of 3D points to the process. The method enables the processing of real world data generated with scanner systems, requiring additional per-point attributes that characterize each measurement. Therefore, common types of scanner systems are analyzed and a general description is developed concerning geometric properties, temporal synchronization, and accuracy.
The surface reconstruction requires fast global operations on data sets with highly dynamic behavior, i.e. an increasing number of points and dynamic mesh changes. This challenge is met by dynamic spatial data structures, the *Dynamic Voxel Space* and the *Extendable Octree*, which are introduced and compared in this thesis. As a further design consideration, the work space is not limited. Hence, suitable data structures must extend themselves, resulting in a computational effort that increases with the size of the covered volume and thus does not meet hard real time conditions. However, a restriction of the object extension limits the complexity, meeting real time requirements. In this work, the *Extendable Octree* is selected, as it outperforms the *Dynamic Voxel Space* for compact data as given at the digitization of single objects, and furthermore preserves the spatial order of the data.
The *RT-SSR* method is verified with simulated data. Typical scan trajectories and varied noise conditions are fed into the system in order to identify a robust set of

parameters. The verification demonstrates the usability of the method for several real world scenarios, e.g. sharp edges, corners, or concavities. Further, the interrelation between the process parameters is discussed and recommendations for an optimal parameter set w.r.t. sensor characteristics are given.
Furthermore, the *RT-SSR* method is integrated into a *visual feedback system* for a manual scanner system, the *DLR Multisensory 3D Modeler*, using the parameter recommendations assessed during the simulations. This *visual feedback system* displays the generated model concurrently to the user. The virtual view is aligned with the scanner pose and the model is augmented with a live stream from the scanner cameras. This feedback system enables an operator to use a manual scanner system intuitively and supports the data acquisition and the inspection of the final model. Further, the software suite integrates methods for texture mapping and object registration. Photorealistic models are generated and automatic registration applications are exemplarily shown. Beyond this visual feedback application, the method is successfully applied for out-of-core processing of huge data sets in the context of automatic 3D model generation of buildings.

8.2 Future work

This thesis extends the field of streaming surface reconstruction from out-of-core applications towards real time reconstruction. However, further extensions and optimizations are envisaged, as outlined in the following.

File swapping strategies and advanced spatial data structures The presented reconstruction method does not require a loading of the entire set of sample points into memory at once, but streams and processes the samples point-by-point. Hence, the processing of real time data as well as huge data sets is possible. However, the reduced sample point set in normal estimation stage and the final mesh are kept in memory, thus limiting the size of the model. In this work, this limitation is bypassed by dividing the data into boxes and separately processing each part, as described in Chapter 7. The limitation can be solved by swapping currently non-involved parts of the data sets in file caches. This extension to the used spatial data structures does not require a complete redesign of the method, as the implemented voxel spaces and octrees can be used directly as core elements of a swapping strategy
Further, if the data only spatially extends in one direction, the octree generates an unnecessary overhead, as it extends uniformly in all directions. An example for this problem is the storage of a 3D map of one level of a building. Here, the map has a predominant expansion in the xy-plane, yet the data structure will also grow in the z-direction. If the data set is additionally too large to fit into memory, smart swapping strategies are required. Here, a combination of Dynamic Voxel Space and octree can be used, i.e. the voxels of the Dynamic Voxel Space contain octrees. Thus, the covered space must not be cubic and a swapping strategy can be limited to voxel level.

Improvement of pose sensing The successful in-the-loop processing of the scanner data highly depends on an accurate pose measurement, as the range images are aligned into the global space in real time and a global minimization of pose errors (e.g. by bundle adjustment) is not possible. Today's pose measurement devices are either of high precision with very restricted working volumes (e.g. CMMs), or are very

flexible in use, yet of low precision (e.g. optical tracking, GPS navigation). Image-based approaches, e.g. ego motion tracking or real-time registration, are precise, however, they typically do not measure the pose absolutely. Moreover, they depend on the measured range images or other raw data red from the range sensor. Hence, the measurements drift and loose their reference system as soon as measurement fails. A real time 6D fusion of absolute positioning systems and relative methods would improve the overall accuracy without restricting the working volume.

Merging multiple sensor streams In this thesis, the generation of a 3D model using a single scanner system is discussed. Beyond this, the fusion of data from multiple scanner systems with different working ranges would be beneficial for some applications. As an example, a historic room could be rapidly sampled with a laser radar. Afterwards, smaller details of the room could be digitized with a manual scanner system. The data could be directly integrated into the existing mesh. This application can principally be performed using the *RT-SSR* method, yet requires further analysis.

A Computational Geometry

A.1 Distance Metrics

The distance metrics between two points or a point and an edge are used to test the elements in a point set or mesh for being inside a ball neighborhood. The respective neighborhoods are defined in Chapter 3.1.

Distance between two points The unsigned distance between two points or vertices $\mathbf{p}$ and $\mathbf{q}$ is given by

$$d_p(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| \quad \mathbf{p}, \mathbf{q} \in \mathbb{R}^3 \tag{A.1}$$

with the euclidean norm

$$\|\mathbf{x}\| = \left\|(x_1\, x_2\, x_3)^T\right\| = \sqrt{x_1^2 + x_2^2 + x_3^2}\ .$$

The euclidean norm is also known as euclidean distance or 2-norm.

Distance between point and edge The unsigned distance between an edge $\mathbf{e}$ and a point $\mathbf{p}$ is the shortest distance between the line segment $\overline{\mathbf{e}_1\mathbf{e}_2}$ and the point $\mathbf{p}$.
Let g denote the line through the points $\mathbf{e}_1$ and $\mathbf{e}_2$, defined by

$$g : \mathbf{x} = \mathbf{e}_1 + \lambda\,(\mathbf{e}_2 - \mathbf{e}_1) \quad \lambda \in \mathbb{R}\ .$$

Here, the scalar λ represents the position of the point $\mathbf{x}$ on the line. For the range $\lambda \in [0, 1]$, the line represents the edge e. Further, let $\lambda_\mathbf{p}$ denote the position of the intersection between the line and the perpendicular through $\mathbf{p}$, given by

$$\lambda_\mathbf{p} = \frac{\langle(\mathbf{p} - \mathbf{e}_1), (\mathbf{e}_2 - \mathbf{e}_1)\rangle}{\|\mathbf{e}_2 - \mathbf{e}_1)\|^2}\ .$$

Hence, the shortest unsigned distance between the edge $\mathbf{e}$ and the point $\mathbf{p}$ is defined by

$$d_e(\mathbf{p}, \mathbf{e}) = \begin{cases} \frac{\|(\mathbf{p}-\mathbf{e}_1)\times(\mathbf{e}_2-\mathbf{e}_1)\|}{\|\mathbf{e}_2-\mathbf{e}_1\|} & 0 < \lambda_\mathbf{p} < 1 \\ d_p(\mathbf{p}, \mathbf{e}_1) & \lambda_\mathbf{p} \leq 0 \\ d_p(\mathbf{p}, \mathbf{e}_2) & \lambda_\mathbf{p} \geq 1 \end{cases}\ . \tag{A.2}$$

A.2 Description of general rotations in 3D space

In Chapter 2.3, the measured scanner pose is denoted as homogeneous 4×4-matrices that consist of a 3×3 rotation sub-matrix $\mathbf{R}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$, known as *rigid motion transformation*. The notation of transformations as homogeneous 4×4-matrices is commonly used in the field of robotics and computer graphics. Generally, a *rigid motion transformation* describes the relation between two coordinate systems or frames. Detailed information concerning transformations and homogeneous coordinates are e.g. given by Craig [Cra89] or Akenine-Moeller [AMH02] and are used within the *OpenGL*-Interface [OSW+04].
The rotation matrix $\mathbf{R}$ describes the rotational part of a transformation. It is an orthogonal matrix whose determinant is equal to one:

$$\mathbf{R}^T = \mathbf{R}^{-1} \quad \det \mathbf{R} = 1 \ .$$

In the following, two representations of rotations beside the matrix notation are introduced and the mapping to the matrix form is described.

A.2.1 Axis-Angle Representation

The *axis-angle* representation, i.e. the rotation about an arbitrary axis, is commonly used in the field of robotics and computer graphics. In Chapter 2.3 the representation is used, for linear interpolation between the frames of two consecutive pose measurements. Here, first the transformation between the two coordinate frames is calculated and then the rotational part is transformed to the *axis-angle* representation. The angle is increased stepwise, in order to calculate frames between the two pose measurements, while the axis remains unchanged. The mapping between rotation matrix and *axis-angle* representation has been described in several ways. Here, the definition of the *OpenGL Architecture Review Board* [OSW+04] is used and described in the following.
Let $\mathbf{a} = (a_x a_y a_z)^T$ be an arbitrary direction vector in $\mathbb{R}^3$ with $\|\mathbf{a}\| = 1$ and let $\alpha \in [0°, 90°]$ denote the angle value that a coordinate system is rotated around $\mathbf{a}$. Further, let

$$\mathbf{S} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \ .$$

The rotation matrix of the pair $(\mathbf{a}, \alpha)$, i.e. the mapping

$$(\mathbf{a}, \alpha) \mapsto \mathbf{R} \ ,$$

is given by

$$\mathbf{R}_{aa}(\mathbf{a}, \alpha) = \mathbf{a}\mathbf{a}^T + \cos\alpha \, (\mathbf{I} - \mathbf{a}\mathbf{a}^T) + \sin\alpha \, \mathbf{S} \ .$$

The inverse mapping

$$\mathbf{R} \mapsto (\mathbf{a}, \alpha)$$

can be calculated by the following rule:
Let $\mathbf{R}$ denote a valid rotation matrix with

$$\mathbf{R} = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix} \ .$$

Further, let $\mathbf{r}$ denote the vector

$$\mathbf{r} = \begin{pmatrix} r_{21} - r_{12} \\ r_{02} - r_{20} \\ r_{10} - r_{01} \end{pmatrix} .$$

Hence, the angle α is given by

$$\alpha = \operatorname{atan2}(\|\mathbf{r}\|, \operatorname{trace}\mathbf{R} - 1) ,$$

and the corresponding direction vector $\mathbf{a}$ by

$$\mathbf{a} = \begin{cases} (001)^T & \alpha = 0 \\ \frac{\mathbf{r}}{2\sin\alpha} & \text{otherwise} \end{cases} .$$

A.2.2 Roll-Pitch-Yaw Representation

A rotation in $\mathbb{R}^3$ can also be represented by the concatenation of three elementary rotations. In Chapter 2.4, the *roll-pitch-yaw* or *x-y-z fixed angles* notation, as e.g. described by Craig [Cra89], is used for defining noise w.r.t. the local sensor coordinate system, i.e. each of the three rotations takes place around an axis in the fixed reference frame.
Let α_r, α_p, and α_y denote the three angles for roll, yaw, and pitch and let $\mathbf{R}_{rpy}(\alpha_r, \alpha_p, \alpha_y)$ denote the corresponding rotation matrix. The mapping between both representations is given by:

$$\mathbf{R}_{rpy}(\alpha_r, \alpha_p, \alpha_y) = \begin{pmatrix} \cos\alpha_r & -\sin\alpha_r & 0 \\ \sin\alpha_r & \cos\alpha_r & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha_p & 0 & \sin\alpha_p \\ 0 & 1 & 0 \\ -\sin\alpha_p & 0 & \cos\alpha_p \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_y & -\sin\alpha_y \\ 0 & \sin\alpha_y & \cos\alpha_y \end{pmatrix} .$$

A detailed description and derivation of the *x-y-z fixed angles* notation is provided by Craig [Cra89].

A.3 Intersection of Edges

In this section, the intersection test between a candidate edge e and an edge of the mesh $e_{\mathcal{L}}$ in $\mathbb{R}^2$, as required at the triangulation update in Chapter 3.5, is described. here, it is not require to test all intersection cases, as not all cases are possible within the triangulation update.
Let the candidate edge e denote the line segment starting at the origin and ending in a point $\mathbf{v}$

$$e = \overline{\mathbf{0v}} \quad \mathbf{0}, \mathbf{v} \in \mathbb{R}^2 .$$

Further, let the edge $e_{\mathcal{L}}$ denote the line segment

$$e_{\mathcal{L}} = \overline{\mathbf{v}_{\mathcal{L}0}\mathbf{v}_{\mathcal{L}1}} \quad \mathbf{v}_{\mathcal{L}0}, \mathbf{v}_{\mathcal{L}1} \in \mathbb{R}^2 .$$

In the context of the triangulation update, the points $\mathbf{v}_{\mathcal{L}0}$ and $\mathbf{v}_{\mathcal{L}1}$ can not be not located at the origin, i.e.

$$\mathbf{v}_{\mathcal{L}0} \neq \mathbf{0} \wedge \mathbf{v}_{\mathcal{L}1} \neq \mathbf{0}$$

If the candidate edge e intersects with the edge $e_{\mathcal{L}}$ in one of its end-points, i.e. if they share the same point

$$\mathbf{v} = \mathbf{v}_{\mathcal{L}0} \vee \mathbf{v} = \mathbf{v}_{\mathcal{L}1} ,$$

the candidate edge satisfies the terms defined in Chapter 3.1 and thus, the edges do not intersect. Otherwise, the edges have to be further tested. Here, the method of *signed triangle area*, as presented in the textbook of O'Rourke [O'R98], is used.
Let $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ denote the vertices of a triangle and let $A(\mathbf{a}, \mathbf{b}, \mathbf{c})$ denote the signed area of the triangle, defined by

$$2A(\mathbf{a}, \mathbf{b}, \mathbf{c}) = (b_0 - a_0)(c_1 - a_1) - (c_0 - a_0)(b_1 - a_1) \ .$$

The point $\mathbf{c}$ is left of the segment $\overline{\mathbf{ab}}$, if the area A_{abc} is greater than zero.
Consequently, the edge e and the edge $e_{\mathcal{L}}$ do not intersect, if both points of one edge are on the same side of the other edge. In Alg. 6 the complete algorithm is visualized. A detailed description of more segment intersection methods is described by O'Rourke [O'R98].

Algorithm 6 Intersection between the edge e and the edge $e_{\mathcal{L}}$.

Require: $e = \overline{\mathbf{0v}}$ and $e_{\mathcal{L}} = \overline{\mathbf{v}_{\mathcal{L}0}\mathbf{v}_{\mathcal{L}0}}$
  if $\mathbf{v} = \mathbf{v}_{\mathcal{L}0}$ **or** $\mathbf{v} = \mathbf{v}_{\mathcal{L}1}$ **then**
    return false
  end if
  if $A(\mathbf{0}, \mathbf{v}, \mathbf{v}_{\mathcal{L}0}) > 0$ **xor** $A(\mathbf{0}, \mathbf{v}, \mathbf{v}_{\mathcal{L}1}) > 0$ **then**
    /* $e_{\mathcal{L}}$ are on the same side of e */
    return false
  end if
  if $A(\mathbf{v}_{\mathcal{L}0}, \mathbf{v}_{\mathcal{L}1,\mathbf{0}}) > 0$ **xor** $A(\mathbf{v}_{\mathcal{L}0}, \mathbf{v}_{\mathcal{L}1,\mathbf{v}}) > 0$ **then**
    /* e is left or right of $e_{\mathcal{L}}$ */
    return false
  end if
  /* e and $e_{\mathcal{L}}$ intersect */
  return true

B

Calculation of Reference Sample Density

In this chapter, the calculations of the formulas Equations (2.13)-(2.16) for determining the expected reference sample density δ are shown. The formulas are the type specific calculation rules for each type of image coordinate system, as introduced in Chapter 2.2.1.

In detail, the simplification of the inverses of the directional sample densities δ_u and δ_v (see Equation (2.11) and (2.12))

$$\begin{aligned}\delta_u^{-1} &= \|\mathbf{p}_{i+1,j} - \mathbf{p}_{i,j}\| \\ \delta_v^{-1} &= \|\mathbf{p}_{i,j+1} - \mathbf{p}_{i,j}\|\end{aligned}$$

to the type-specific Equations (2.13)-(2.16) are explained. Here, it is assumed that both points $\mathbf{p}_{i,j+1}$ and $\mathbf{p}_{i,j}$ origin from distance pixels with a value of $d_{i,j+1} = d_{i,j} = d$.

B.1 Helper Equations

In the following a set of conversions as used in the following sections are delineated. The grid coordinates $u(i)$ and $v(j)$, as introduced in Chapter 2.2.1, are the physical coordinates of a pixel d_{ij} at the i-th row and j-th column in the native image coordinate system. They are described by the offsets u_0, v_0 and the sample widths $\Delta u, \Delta v$,

$$\begin{aligned}u(i) &= u_0 + i \cdot \Delta u; \quad i \in \mathbb{N}_m \\ v(j) &= v_0 + j \cdot \Delta v; \quad j \in \mathbb{N}_n \ .\end{aligned}$$

Hence, the differences of the grid coordinates can be simplified by

$$u(i+1) - u(i) = \Delta u \tag{B.1}$$
$$v(j+1) - v(j) = \Delta v \ . \tag{B.2}$$

If a grid coordinate describes an angular value, the sinusoid differences are required. They can be simplified by

$$\sin u(i+1) - \sin u(i) = 2\cos(u(i+1)+u(i))\sin\frac{\Delta u}{2} \tag{B.3}$$

$$\cos u(i+1) - \cos u(i) = -2\sin(u(i+1)+u(i))\sin\frac{\Delta u}{2} \tag{B.4}$$

$$\sin v(j+1) - \sin v(j) = 2\cos(v(j+1)+v(j))\sin\frac{\Delta v}{2} \tag{B.5}$$

$$\cos v(j+1) - \cos v(j) = -2\sin(v(j+1)+v(j))\sin\frac{\Delta v}{2}\ . \tag{B.6}$$

B.2 Cartesian Geometry

A sample point $\mathbf{p}_{ij}$ in a Cartesian coordinate system is described by

$$\mathbf{p}_{ij} = \begin{pmatrix} u \\ v \\ d \end{pmatrix}\ .$$

The corresponding inverse reference density δ_u^{-1} in u-direction is simplified by

$$\begin{aligned} \delta_u^{-1} &= \left\| \begin{pmatrix} u(i+1) \\ v(j) \\ d \end{pmatrix} - \begin{pmatrix} u(i) \\ v(j) \\ d \end{pmatrix} \right\| \\ &= \left\| \begin{matrix} \Delta u \\ 0 \\ 0 \end{matrix} \right\| && \text{(applying Equation (B.1))} \\ &= |\Delta u|\,, \end{aligned}$$

and the inverse reference density δ_v^{-1} in v-direction is simplified by

$$\begin{aligned} \delta_v^{-1} &= \left\| \begin{pmatrix} u(i) \\ v(j+1) \\ d \end{pmatrix} - \begin{pmatrix} u(i) \\ v(j) \\ d \end{pmatrix} \right\| \\ &= \left\| \begin{matrix} 0 \\ \Delta v \\ 0 \end{matrix} \right\| && \text{(applying Equation (B.2))} \\ &= |\Delta v|\,. \end{aligned}$$

B.3 Perspective Geometry

A sample point $\mathbf{p}_{ij}$ in a perspecitve coordinate system is described by

$$\mathbf{p}_{ij} = \begin{pmatrix} d \cdot u \\ d \cdot v \\ d \end{pmatrix}\ .$$

The corresponding inverse reference density δ_u^{-1} in u-direction is simplified by

$$\begin{aligned}
\delta_u^{-1} &= \left\| \begin{pmatrix} du(i+1) \\ dv(j) \\ d \end{pmatrix} - \begin{pmatrix} du(i) \\ dv(j) \\ d \end{pmatrix} \right\| \\
&= \left\| \begin{matrix} d\Delta u \\ 0 \\ 0 \end{matrix} \right\| && \text{(applying Equation (B.1))} \\
&= d|\Delta u| \,,
\end{aligned}$$

and the inverse reference density δ_v^{-1} in v-direction is simplified by

$$\begin{aligned}
\delta_v^{-1} &= \left\| \begin{pmatrix} du(i) \\ dv(j+1) \\ d \end{pmatrix} - \begin{pmatrix} du(i) \\ dv(j) \\ d \end{pmatrix} \right\| \\
&= \left\| \begin{matrix} 0 \\ d\Delta v \\ 0 \end{matrix} \right\| && \text{(applying Equation (B.2))} \\
&= d|\Delta v| \,.
\end{aligned}$$

B.4 Cylindrical Geometry

A sample point $\mathbf{p}_{ij}$ in a cylindrical coordinate system is described by

$$\mathbf{p}_{ij} = \begin{pmatrix} d \cdot \sin u \\ v \\ d \cdot \cos u \end{pmatrix} \,.$$

The corresponding inverse reference density δ_u^{-1} in u-direction is simplified by

$$\begin{aligned}
\delta_u^{-1} &= \left\| \begin{pmatrix} d\sin u(i+1) \\ v(j) \\ d\cos u(i+1) \end{pmatrix} - \begin{pmatrix} d\sin u(i) \\ v(j) \\ d\cos u(i) \end{pmatrix} \right\| \\
&= \left\| d \begin{pmatrix} \sin u(i+1) - \sin u(i) \\ 0 \\ \cos u(i+1) - \cos u(i) \end{pmatrix} \right\| \\
&= \left\| d \begin{pmatrix} 2\cos(u(i+1)+u(i))\sin\frac{\Delta u}{2} \\ 0 \\ -2\sin(u(i+1)+u(i))\sin\frac{\Delta u}{2} \end{pmatrix} \right\| && \text{(applying Eq. (B.3) and (B.4))} \\
&= d\left[(2\cos(u(i+1)+u(i))\sin\tfrac{\Delta u}{2})^2\right. \\
&\quad \left. + (-2\sin(u(i+1)+u(i))\sin\tfrac{\Delta u}{2})^2\right]^{\frac{1}{2}} \\
&= 2d\sin\tfrac{|\Delta u|}{2}\left[\cos^2(u(i+1)+u(i)) + \sin^2(u(i+1)+u(i))\right]^{\frac{1}{2}} \\
&= 2d|\sin\tfrac{\Delta u}{2}| \,,
\end{aligned}$$

and the inverse reference density δ_v^{-1} in v-direction is simplified by

$$\begin{aligned}
\delta_v^{-1} &= \left\| \begin{pmatrix} d\sin u(i) \\ v(j+1) \\ d\cos u(i) \end{pmatrix} - \begin{pmatrix} d\sin u(i) \\ v(j) \\ d\cos u(i) \end{pmatrix} \right\| \\
&= \left\| \begin{matrix} 0 \\ \Delta v \\ 0 \end{matrix} \right\| && \text{(applying Equation (B.2))} \\
&= |\Delta v| \,.
\end{aligned}$$

B.5 Spherical Geometry

A sample point $\mathbf{p}_{ij}$ in a spherical coordinate system (with (u, v)-order) is described by

$$\mathbf{p}_{ij} = \begin{pmatrix} d \cdot \sin u \cos v \\ d \cdot \sin u \sin v \\ d \cdot \cos u \end{pmatrix} .$$

The corresponding inverse reference density δ_u^{-1} in u-direction is simplified by

$$\begin{aligned}
\delta_u^{-1} &= \left\| \begin{pmatrix} d\sin u(i+1)\cos v(j) \\ d\sin u(i+1)\sin v(j) \\ d\cos u(i+1) \end{pmatrix} - \begin{pmatrix} d\sin u(i)\cos v(j) \\ d\sin u(i)\sin v(j) \\ d\cos u(i) \end{pmatrix} \right\| \\
&= \left\| d \begin{pmatrix} (\sin u(i+1) - \sin u(i))\cos v(j) \\ (\sin u(i+1) - \sin u(i))\sin v(j) \\ \cos u(i+1) - \cos u(i) \end{pmatrix} \right\| \text{(applying Eq. (B.3) and (B.4))} \\
&= d \left[(\sin u(i+1) - \sin u(i))^2 + (\cos u(i+1) - \cos u(i))^2 \right]^{\frac{1}{2}} \\
&= d \left[4\cos^2(u(i+1)+u(i))\sin^2\tfrac{\Delta u}{2} + \sin^2(u(i+1)+u(i))\sin^2\tfrac{\Delta u}{2} \right]^{\frac{1}{2}} \\
&= 2d|\sin\tfrac{\Delta u}{2}|
\end{aligned}$$

and the inverse reference density δ_v^{-1} in v-direction is simplified by

$$\begin{aligned}
\delta_v^{-1} &= \left\| \begin{pmatrix} d\sin u(i)\cos v(j+1) \\ d\sin u(i)\sin v(j+1) \\ d\cos u(i) \end{pmatrix} - \begin{pmatrix} d\sin u(i)\cos v(j) \\ d\sin u(i)\sin v(j) \\ d\cos u(i) \end{pmatrix} \right\| \\
&= d \left\| \sin u(i) \begin{pmatrix} \cos v(j+1) - \cos v(j) \\ \sin v(j+1) - \sin v(j) \\ 0 \end{pmatrix} \right\| \text{(applying Eq. (B.5) and (B.6))} \\
&= d\,|\sin u(i)|\,\left[(\cos v(j+1) - \cos v(j))^2 + (\sin v(j+1) - \sin v(j))^2 \right]^{\frac{1}{2}} \\
&= d\,|\sin u(i)|\,\left[4\cos^2(v(j+1)+v(j))\sin^2\tfrac{\Delta v}{2} \right. \\
&\quad \left. +4\sin^2(v(j+1)+v(j))\sin^2\tfrac{\Delta v}{2} \right]^{\frac{1}{2}} \\
&= 2d\,|\sin u(i)|\,|\sin\tfrac{\Delta v}{2}|
\end{aligned}$$

Bibliography

[ABCO+03] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[ACA07] R. Allègre, R. Chaine, and S. Akkouche. A streaming algorithm for surface reconstruction. In *Proceedings of the 5th Eurographics symposium on Geometry processing (SGP)*, pages 79–88, Aire-la-Ville, Switzerland, 2007.

[ACK01] N. Amenta, S. Choi, and R.K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153(27), July 2001.

[AMF03] S. Azernikov, A. Miropolsky, and A. Fischer. Surface reconstruction of freeform objects based on multiresolution volumetric method. In *Proceedings of the 8th ACM symposium on Solid modeling and applications (SM)*, pages 115–126, New York, USA, 2003.

[AMH02] T. Akenine-Moeller and E. Haines. *Real-Time Rendering*. A. K. Peters, 2nd edition, 2002.

[AS00] M. Attene and M. Spagnuolo. Automatic surface reconstruction from point sets in space. *Computer Graphics Forum*, 19(3), August 2000.

[BBX95] Chandrajit Bajaj, Fausto Bernardini, and Guoliang Xu. Adaptive reconstruction of surfaces and scalar fields from dense scattered trivariate data. Computer Science Technical Report CSD-TR-95-028, Department of Computer Sciences, Purdue University, April 1995.

[Ber03] G. Van Den Bergen. *Collision Detection in Interaction 3D Environments*. Morgan Kaufmann, 2003.

[Bes88] P. J. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[BH04] T. Bodenmüller and G. Hirzinger. Online surface reconstruction from unorganized 3d-points for the DLR hand-guided scanner system. In *Proceedings of 2nd Symposium on 3D Data Processing, Visualization, Transmission (3DPVT)*, pages 285–292, Thessaloniki, Greece, September 2004.

[BM92] P. J. Besl and H. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[Boi84] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, 1984.

[BS97] G. Barequet and M. Sharir. Partial surface and volume matching in three dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):929–948, 1997.

[BSSH07] Tim Bodenmüller, Wolfgang Sepp, Michael Suppa, and Gerd Hirzinger. Tackling multi-sensory 3d data acquisition and fusion. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2180–2185, San Diego, USA, October 2007.

[Bur06] D. Burschka. Robust feature correspondences for vision-based navigation with slow frame-rate cameras. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 947–952, Beijing, China, Oktober 2006.

[Cha03] R. Chaine. A geometric convection approach of 3-d reconstruction. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 218–229, Aachen, Germany, June 2003.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 303–312, New Orleans, USA, August 1996.

[Cra89] John J. Craig. *Introduction to Robotics.* Addison-Wesley Publishing Company, 2nd edition, 1989.

[Cur97] Brian Curless. *New methods for surface reconstruction from range images*. PhD thesis, Stanford University, 1997.

[DG03] T. K. Dey and S. Goswami. Tight cocone: a water-tight surface reconstructor. In *Proceedings of the 8th ACM symposium of solid modeling and applications*, pages 127–134, Seattle, USA, June 2003.

[DNRR05] J. Davis, D. Nehab, R. Ramamoorthi, and S. Rusinkiewicz. Spacetime stereo: a unifying framework for depth from triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):296–302, Februar 2005.

[DNZ06] Z. Deng, J. Niu, and J. Zhang. A realistic 3-d reverse modeling system based on real-world sampling dataset. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5327–5332, Beijing, China, October 2006.

[EM94] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[FCOS05] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24(3):544–552, 2005.

[FCS07] V. Fiorin, P. Cignoni, and R. Scopigno. Out-of-core mls reconstruction. In *Proceedings of the 9th IASTED International Conference on Computer Graphics and Imaging (CGIM)*, pages 27–34, Innsbruck, Austria, February 2007.

[FH08] S. Fuchs and G. Hirzinger. Extrinsic and depth calibration of tof-cameras. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, USA, June 2008.

[FR01] M. S. Floater and M. Reimers. Meshless parameterization and surface reconstruction. *Comput. Aided Geometric Design*, 18(2):77–92, 2001.

[GKS00] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum*, 19(3), August 2000. ISSN 1067-7055.

[GMGP05] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Symposium on Geometry Processing*, pages 197–206, Vienna, Austria, July 2005.

[Gop01] M. Gopi. *Theory and practice of sampling and reconstruction for manifolds with boundaries*. PhD thesis, University of North Carolina, 2001.

[Gut84] A. Guttman. A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[Hal05] T. Hales. A proof of the kepler conjecture. *Annals of Mathematics*, 162:1065–1185, 2005.

[HBH+05] G. Hirzinger, T. Bodenmüller, H. Hirschmüller, R. Liu, W. Sepp, M. Suppa, T. Abmayr, and B. Strackenbrock. Photo-realistic 3d modelling - from robotics perception towards cultural heritage. In *International Workshop on Recording, Modeling and Visualization of Cultural Heritage*, Asona, Switzerland, May 2005.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH 92*, Computer Graphics Proceedings, Annual Conference Series, pages 71–78, Chicago, Illinois, July 1992.

[HDH97] F. Hacker, J. Dietrich, and G. Hirzinger. A laser-triangulation based miniaturized 2-d range-scanner as integral part of a multisensory robot-gripper. In *Topical Meeting on Optoelectronic Distance/Displacement Measurements and Applications (ODIMAP)*, 1997.

[HI00] A. Hilton and J. Illingworth. Geometric fusion for a hand-held 3d sensor. *Machine Vision Applications*, 12(1):44–51, 2000.

[Hir03] Heiko Hirschmüller. *Stereo Vision Based Mapping and Immediate Virtual Walkthroughs*. PhD thesis, Montfort University, Leicester, UK, 2003.

[Hir06] H. Hirschmüller. Stereo vision in structured environments by consistent semi-global matching. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2386–2393, New York, USA, June 2006.

[HK07] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis, USA, June 2007.

[Hop94] Hugues Hoppe. *Surface Reconstruction from Unorganized Points*. PhD thesis, University of Washington, 1994.

[IYDT07] I. Ishii, K. Yamamoto, K. Doi, and T. Tsuji. High-speed 3d image acquisition using coded structured light projection. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 925–930, San Diego, USA, October 2007.

[KBB+07] R. Konietschke, A. Busam, T. Bodenmüller, T. Ortmaier, M. Suppa, J. Wiechnik, T. Welzel, G. Eggers, G. Hirzinger, and R. Marmulla. Accuracy identification of markerless registration with the dlr handheld 3d-modeller in medical applications. In *Prodeedings of CURAC 2007*, Karlsruhe, Germany, October 2007.

[KBE+04] W. Korb, T. Bodenmüller, G. Eggers, T. Ortmaier, M. Schneberger, M. Suppa, J. Wiechnik, R. Marmulla, and S. Hassfeld. Surface-based image-to-patient-registration using a hand-guided laser-range scanner system. In *18th International Congress and Exhibition on Computer Assisted Radiology and Surgery (CARS)*, Chicago, USA, June 2004.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 163–169, 1987.

[LCOL07] Y. Lipman, D. Cohen-Or, and D. Levin. Data-dependent mls for faithful surface approximation. In *Proceedings of the fifth Eurographics symposium on Geometry processing (SGP)*, pages 59–67, Barcelona, Spain, July 2007.

[Lev03] D. Levin. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, pages 37–49, 2003.

[LH05] R. Liu and G. Hirzinger. Matching-tree - a novel data structure for global automatic matching. In *7th Conference on Optical 3-D Measurement Techniques*, Vienna, Austria, October 2005.

[Lie03] P. Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP)*, pages 200–205, Aachen, Germany, June 2003.

[LSHB07] R. Liu, M. Suppa, G. Hirzinger, and D. Burschka. Modelling the world in real time. In *8th Conference on Optical 3D Measurement Techniques*, Zurich, Switzerland, July 2007.

[MN03] N. J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the 19sth annual symposium on Computational geometry (SCG)*, pages 322–328, San Diego, USA, June 2003.

[OEF+06] C. Ott, O. Eiberger, W. Friedl, B. Bäuml, U. Hillenbrand, C. Borst, A. Albu-Schäffer, B. Brunner, H. Hirschmüller, S. Kielhöfer, R. Konietschke, M. Suppa, T. Wimböck, F. Zacharias, and G. Hirzinger. A humanoid two-arm system for dexterous manipulation. In *6th IEEE-RAS International Conference on Humanoid Robots*, pages 276 – 283, December 2006.

[O'R98] Josef O'Rourke. *Computational Geometry in C*. Cambridge University Press, 2nd edition, 1998.

[OSW+04] OpenGL Architecture Review Board, D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 1.4*. Addison-Wesley Professional, 4th edition, 2004.

[Paj05] R. Pajarola. Stream-processing points. In *Proceedings of IEEE Visualization (VIS)*, pages 239– 246, Minneapolis, USA, October 2005.

[PGK02] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of IEEE Visualization (VIS)*, pages 163–170, Boston, USA, October 2002.

[PTVF92] William Press, Saul Teukolsky, William Vetterling, and Brian Flannerys. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[RHHL02] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *Proceedings on Computer Graphics (SIGGRAPH)*, San Antonio, Texas, July 2002.

[RL00] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings on Computer Graphics (SIGGRAPH)*, pages 343–352, 2000.

[Rus01] S. Rusinkiewicz. *Real-Time Acquisition and Rendering of large 3D Models*. PhD thesis, Stanford University, 2001.

[Sam90] Hanan Samet. *Applications of spatial data structures: computer graphics, image processing and GIS*. Addison-Wesley, 1990.

[Sch06] Carsten Scheibe. *Algorithms for the Evaluation of Modern Sensors in Close-Range Photogrammetry*. PhD thesis, Georg-August-University Göttingen, 2006.

[Sed92] Robert Sedgewick. *Algorithms in C++*. Addison-Wesley, 1992.

[Sep08] Wolfgang Sepp. *Visual Servoing of Textured Free-Form Objects in 6 Degrees of Freedom*. PhD thesis, Technical University Munich, 2008.

[SH06] K. H. Strobl and G. Hirzinger. Optimal hand-eye calibration. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4647–4653, Beijing, China, October 2006.

[SKL+07] M. Suppa, S. Kielhöfer, J. Langwald, F. Hacker, K. H. Strobl, and G. Hirzinger. The 3d-modeller: A multi-purpose vision platform. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 781–787, Rome, Italy, April 2007.

[SKR+01] K. Scheibe, H. Korsitzky, R. Reulke, M. Scheele, and M. Solbrig. Eyescan - a high resolution digital panoramic camera. In *Robot Vision*, volume 1998/2001 of *Lecture Notes in Computer Science*, pages 77–83. Springer, Berlin, 2001.

[SRR02] W. Scott, G. Roth, and J. F. Rivest. Pose error effects on range sensing. In *Proceedings of Vision Interface (VI)*, page 331, Calgary, Canada, May 2002.

[SSW+04] K.H. Strobl, W. Sepp, E. Wahl, T. Bodenmuller, M. Suppa, J.F. Seara, and G. Hirzinger. The dlr multisensory hand-guided device: the laser stripe profiler. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1927–1932Vol.2, Apr 26-May 1, 2004.

[Sup07] Michael Suppa. *Autonomous Robot Work Cell Exploration Using Multisensory Eye-in-Hand Systems*. PhD thesis, University Hannover, 2007.

[TKH07] J. Takei, S. Kagami, and K. Hashimoto. 3,000-fps 3-d shape measurement using a high-speed camera-projector system. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3211–3216, San Diego, USA, October 2007.

[TL94] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH 94*, pages 311–318, Orlando, Florida, July 1994.

[WLG07] T. Weise, B. Leibe, and L. Van Gools. Fast 3d scanning with automatic motion compensation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis, USA, June 2007.

[WMW06] S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *Proceedings of 28th Annual Symposium of the German Association for Pattern Recognition (DAGM)*, pages 718–728, Berlin, Germany, September 2006.

[WOK05] J. Wang, M. M. Oliveira, and A. E. Kaufman. Reconstructing manifold and non-manifold surfaces from point clouds. In *Proceedings of IEEE Visualization (VIS)*, pages 415–422, Minneapolis, USA, October 2005.

[YBD+07] M. Young, E. Beeson, J. Davis, S. Rusinkiewicz, and R. Ramamoorthi. Viewpoint-coded structured light. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, USA, June 2007.

[ZSK06] C. Zach, M. Sormann, and K. Karner. High-performance multi-view reconstruction. In *Proceedings of 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, pages 113–120, North Carolina, USA, June 2006.