

DISS. ETH NO. 21009

Multi-Robot Coverage and Path Planning for the Inspection of Curved Surfaces

A dissertation submitted to
ETH ZURICH
for the degree of
Doctor of Sciences

presented by

Andreas Breitenmoser

Master of Science in
Electrical Engineering and Information Technology
Eidgenössische Technische Hochschule (ETH) Zurich, Switzerland

born on May 12, 1983
citizen of Mosnang SG, Switzerland

accepted on the recommendation of

Prof. Dr. Roland Siegwart,	ETH Zurich
Prof. Dr. Alcherio Martinoli,	EPF Lausanne
Dr. Roland Moser,	ALSTOM
Prof. Dr. Daniela Rus,	Massachusetts Institute of Technology

2013

*To all the good people I knew but are gone,
and to my little sister I have never known.*

Acknowledgments

First and foremost, I would like to thank my thesis supervisor, Prof. Roland Siegwart, for providing me with all the necessary support to realize this work and for having me as a PhD student at the Autonomous Systems Laboratory (ASL)—definitely a great place to do robotics research, to learn about new crazy robot designs and to find a friendly atmosphere.

I am also very grateful that I had the chance to spend three unforgettable research visits at the Distributed Robotics Laboratory (DRL) at MIT. I would like to thank Prof. Daniela Rus for hosting me, for all the inspiration and guidance. I wish to thank Prof. Alcherio Martinoli for the motivating discussions whenever we met at conferences or other occasions during the past years. I appreciate Dr. Roland Moser’s vision for robotic inspection, which initiated this research project. The research was supported by ALSTOM. I am honored that Prof. Daniela Rus, Prof. Alcherio Martinoli and Dr. Roland Moser agreed to be on my doctoral committee.

This work would not have been possible without the support of various people. I wish to thank the following colleagues: Javier Alonso-Mora, Philipp Krüsi, Dr. Martin Rufli and Ulrich Schwesinger for stimulating discussions about robot motion planning, coordination and control; Prof. Mac Schwager, Dr. Seungkook Yun, Byoungkwon An and Prof. Eduardo Torres-Jara for shared insights and a warm reception; all the colleagues at ASL and DRL I spent a good time with at work, at conferences or other lab activities.

Many thanks to all the great co-authors for the fruitful collaboration. I would like to thank Dr. Gilles Caprari, Dr. Fabien Tâche, Dr. Wolfgang Fischer, Dr. Laurent Kneip, François Pomerleau and Prof. Cédric Pradalier for the help on MagneBike, Dominik Haumann for the DisCoverage extension, Hannes Sommer for incomparable mathematical excursions, and Javier Alonso-Mora, Dr. Martin Rufli and Dr. Paul Beardsley for the joint work on the wonderful DisplaySwarm alias Pixelbots project.

During my doctorate, I also had the opportunity to work together with many graduate and undergraduate students who were eager to explore new

ideas. Especially, I wish to thank Christian Forster, Elena Stumm and Jean-Claude Metzger. Elena's and my joint (sometimes late night and remote) testing of the MagneBike robots, as well as Jean-Claude's company at work (and when exploring Boston) are in good memory.

Furthermore, I would like to thank the technical support staff of ASL, Thomas Baumgartner, Stefan Bertschi, Markus Bühler and Dario Fenner, for their help in building and maintaining our robots at an exceptionally high level of quality.

Finally, I wish to thank my friends and my family for all their support. I sincerely thank my parents—for everything.

Zurich, January 2013

Abstract

This dissertation proposes different algorithms for multi-robot coverage and path planning in 3D environments of challenging geometry. The problem of deploying a group of mobile robots is studied for covering environments which present obstacles, nonconvexities and curved surfaces.

The general motivation of this work is drawn from *robotic inspection*. The periodic inspection of industrial structures and infrastructure is crucial for their safe and efficient future operation. The *inspection task* can be understood as the *complete coverage* of a structure in order to detect developing defects and prevent more severe damages, destruction or supply shortfalls. In seeking to develop *assistive inspection tools* of increased autonomy, autonomous inspection robots, equipped with appropriate coverage and path planning algorithms, could lead to desired improvements in reliability, efficiency, accessibility and flexibility of today's inspection and maintenance.

This dissertation is structured in three parts, of which the first part embeds the latter two parts, which present the thesis's main contributions of multi-robot coverage and path planning.

The first part describes the motivating inspection scenario and derives the underlying problem formulation for our work. The *hybrid coverage concept* is introduced, which provides a basic formulation for multi-robot coverage: hybrid coverage combines deployment and sweeping motion by performing one after the other. *Hybrid Voronoi coverage* offers particularly interesting properties in the context of the inspection scenario. The hybrid Voronoi coverage solution deploys the robots by constructing a Voronoi tessellation of the environment, assigning each robot to a Voronoi region and steering the robots to the regions' centroids in a first stage. In a second stage, each robot covers its assigned Voronoi region by moving along a sweeping pattern.

The second part then develops the methods that allow for deploying a group of robots in *nonconvex environments* and on *curved surfaces* within the first stage of hybrid Voronoi coverage. To cope with nonconvex environments, a coverage solution is designed in continuous space, which combines

Voronoi coverage with *path planning*. The robots explore obstacle boundaries and plan paths to circumnavigate obstacles on the way to their goal positions. If a goal position is not reachable, a robot moves to the closest accessible position in free space, which realizes a *gradient projection method* in distributed fashion. In addition, two coverage solutions for curved surfaces, using a discrete environment representation, are presented. Each robot models the environment as triangle mesh, which provides a *graph embedded in 3D space*. The first coverage algorithm computes shortest path distances and propagates a discrete wavefront on the robot’s graph representation to obtain a centroidal graph Voronoi tessellation. The second coverage algorithm approximates distances on the surface through the Euclidean distance in the ambient space and locally exchanges vertices between adjacent Voronoi regions of robot neighbors to create a final centroidal Voronoi configuration. By additionally computing a metric tensor field locally on the surface, the shape of the Voronoi regions can be made *adaptive* in position, size, orientation and aspect ratio according to the present anisotropy.

The third part addresses the underlying problem of moving individual robots through uneven terrain and over curved surfaces of the 3D environments. *Mesh-based* and *point-based* path planning are proposed. The presented navigation solution for mesh-based path planning generates a *triangle mesh* from 3D point clouds and plans a *triangle strip path* on the mesh. A vector field is created along the extracted strip, which guides a robot over the approximated surface. Point clouds, unlike triangle meshes, intrinsically do not provide connectivity information. Point-based path planning saves the mesh generation step but needs ways to *establish connectivity* throughout the point cloud. Two navigation solutions for point-based path planning are suggested. The first navigation solution for *dense point cloud planning* generates a dense point cloud in a preprocessing step, using *tensor voting* for augmenting the point cloud. Connectivity is established by subdividing the augmented point cloud into a regular grid structure and constructing a graph *incrementally*. A specialized graph-based planner connects the successive discretized robot poses along the surface into a *6-DoF path*, considering kinematic as well as structural constraints. The 6-DoF path is then transformed from 3D space into 2D space by projecting movements into local tangent planes of the surface, which makes 2D trajectory tracking for robot control possible. The second navigation solution for *sparse point cloud planning* fits *quadric surface patches* locally to the augmented 3D point cloud to establish connectivity. By developing and expanding *motion primitives* on the surface patches, the robot is finally controlled on the surface.

The coverage and navigation solutions have been evaluated in simulations, and in experiments with the *e-puck robot* and *MagneBike robot* platforms. MagneBikes are compact inspection robots, which can climb ferromagnetic structures due to their magnetic wheels. With respect to the multi-robot inspection task, a vision-based relative localization system, consisting of a *camera* and a *target module* with four active or passive visual markers has been developed and evaluated. It provides a lightweight solution for localizing a robot's full *6D pose*, which is especially beneficial for the relative localization of inspection robots that climb curved surfaces in 3D space.

In summary, the *coverage* and *navigation solutions* developed within this thesis investigate and contribute concepts that are of general relevance for environment modeling, path planning and coordination of robots operating in real world scenarios.

Keywords: Multi-robot systems, Robot deployment, Distributed algorithms, Voronoi coverage, Coverage path planning, Path planning in 3D environments, Environment modeling, LIDAR point clouds, Mesh generation, Curved surfaces, Wheeled and climbing robots, Relative robot localization, Robotic inspection

Kurzfassung

Diese Dissertation behandelt verschiedene Algorithmen für die Flächenabdeckung durch Mehr-Roboter-Systeme und deren Pfadplanung in geometrisch anspruchsvollen 3D-Umgebungen. Es wird die Problemstellung untersucht, wie mehrere Roboter zu verteilen sind, um Umgebungen abzudecken, die Hindernisse, nichtkonvexe und gekrümmte Oberflächen aufweisen.

Der Beweggrund für diese Arbeit ist in der *Inspektion mit Robotern* begründet. Periodische Inspektionen von Infrastrukturbauten und industriellen Anlagen sind entscheidend für deren sicheren und effizienten Betrieb. Die *Inspektionsaufgabe* kann durch die *vollständige Oberflächenabdeckung* einer Struktur gelöst werden, wobei entstehende Defekte entdeckt und somit ernste Schäden, Zerstörung oder Versorgungsengpässe vermieden werden können. Wird die Selbständigkeit der *unterstützenden Inspektionswerkzeuge* erhöht und weiterentwickelt, könnten autonome Inspektionsroboter, die mit geeigneten Algorithmen für die Flächenabdeckung und Pfadplanung ausgerüstet sind, zu wünschenswerten Verbesserungen bezüglich Zuverlässigkeit, Effizienz, Zugänglichkeit und Flexibilität bei Inspektion und Unterhalt beitragen.

Die Dissertation ist in drei Teile gegliedert. Der erste Teil führt auf die beiden folgenden Teile hin, in denen die Hauptbeiträge der Flächenabdeckung durch Mehr-Roboter-Systeme und deren Pfadplanung behandelt werden.

Der erste Teil beschreibt das herausfordernde Inspektionsumfeld und leitet die zugrunde liegenden Fragestellungen für unsere Arbeit ab. Das *hybride Flächenabdeckungskonzept* wird eingeführt, das eine grundlegende Formulierung für die Flächenabdeckung durch Mehr-Roboter-Systeme bereitstellt: Die hybride Flächenabdeckung kombiniert die Formierung und die Flächen überstreifende Bewegung durch Ausführen eines Schrittes nach dem andern. Die *hybride Voronoi-Flächenabdeckung* weist besonders interessante Eigenschaften im Zusammenhang mit dem Inspektionsumfeld auf. Die Lösung mit der hybriden Voronoi-Flächenabdeckung formiert die Roboter durch eine Voronoi-Flächeneinteilung der Umgebung, indem jeder Roboter einem Voro-

noi-Bereich zugeordnet wird. Jeder Roboter wird zu Beginn in den gewichteten Flächenschwerpunkt eines Bereiches gesteuert. Anschliessend deckt jeder Roboter seinen zugeordneten Voronoi-Bereich ab, indem er sich entlang eines Flächen überstreifenden Musters bewegt.

Der zweite Teil entwickelt die Methoden, die es erlauben, eine Gruppe von Robotern in *nichtkonvexen Umgebungen* und auf *gekrümmten Oberflächen* im ersten Schritt der hybriden Voronoi-Flächenabdeckung zu positionieren. Um nichtkonvexe Umgebungen zu meistern, wird für die Flächenabdeckung im kontinuierlichen Raum eine Lösung entworfen, die die Voronoi-Flächenabdeckung mit der *Pfadplanung* kombiniert. Die Roboter erkunden die Grenzen von Hindernissen und planen Wege, um Hindernisse auf dem Weg zu ihren Zielpositionen zu umgehen. Wenn eine Zielposition nicht erreichbar ist, bewegt sich der Roboter zum nächstgelegenen erreichbaren Punkt im Raum. Dies verwirklicht ein *Gradienten-Projektionsverfahren* auf verteilte Art und Weise. Zudem werden zwei Lösungen für die Flächenabdeckung von gekrümmten Oberflächen mit unterschiedlichen Umgebungsdarstellungen vorgestellt. Jeder Roboter modelliert die Umgebung als Dreiecksgitternetz, welches einen *im 3D-Raum eingebetteten Graphen* bereitstellt. Der erste Flächenabdeckungsalgorithmus berechnet die kürzesten Wegstrecken und breitet eine diskrete Wellenfront auf der Graph-Darstellung des Roboters aus, um eine Flächenschwerpunkt basierte Graphen-Voronoi-Flächeneinteilung zu erzielen. Der zweite Flächenabdeckungsalgorithmus nähert Distanzen auf der Oberfläche durch die Euklidische Distanz im umgebenden Raum an und tauscht angrenzende Knoten zwischen Voronoi-Bereichen benachbarter Roboter lokal aus, um schliesslich eine Flächenschwerpunkt basierte Voronoi-Anordnung zu erstellen. Indem lokal auf der Oberfläche ergänzend ein metrisches Tensorfeld berechnet wird, kann die Form der Voronoi-Bereiche entsprechend der gegenwärtigen Anisotropie in der Position, der Grösse, der Orientierung und dem Abbildungsverhältnis *angepasst* werden.

Der dritte Teil befasst sich mit der zugrunde liegenden Fragestellung, wie einzelne Roboter durch unebenes Gelände und über gekrümmte Oberflächen von 3D-Umgebungen zu bewegen sind. Eine *Gitternetz* und *Punkt basierte* Pfadplanung werden vorgeschlagen. Die vorgestellte Navigationslösung für die Gitternetz basierte Pfadplanung erstellt ein *Dreiecksgitternetz* aus 3D-Punktwolken und plant einen *aus Dreiecken bestehenden Streifenpfad* auf dem Gitternetz. Entlang des resultierenden Streifens wird ein Vektorfeld erstellt, das den Roboter über die angenäherte Oberfläche führt. Im Gegensatz zu Dreiecksgitternetzen enthalten Punktwolken keine Information über die Verbindungen von Punkten. Die Punkt basierte Pfadplanung spart den Gitternetz-Erstellungsschritt ein, muss aber *Verbindungen* durch die Punkt-

wolke aufbauen. Zwei Navigationslösungen für die Punkt basierte Pfadplanung werden angeregt. Die erste Lösung für die *Planung mit dichten Punktwolken* generiert eine dichte Punktwolke in einem Vorverarbeitungsschritt, indem zusätzliche Information zur Punktwolke durch *Tensor Voting* dazugefügt wird. Verbindungen werden aufgebaut, indem die mit zusätzlicher Information versehene Punktwolke in eine regelmässige Gitterstruktur unterteilt und *schrittweise* ein Graph konstruiert wird. Ein spezialisierter, Graph basierter Planer verbindet laufend die diskretisierten Roboterpositionen entlang der Oberfläche zu einem *6-DoF Pfad*, während er kinematische und strukturelle Randbedingungen beachtet. Der 6-DoF Pfad wird anschliessend vom 3D-Raum in den 2D-Raum transformiert, indem Bewegungen in lokalen Tangentialebenen der Oberfläche abgebildet werden. Dies ermöglicht den Robotern das Verfolgen von Trajektorien im 2D-Raum. Die zweite Navigationslösung für die *Planung mit spärlichen Punktwolken* passt *Quadriken* lokal an die 3D-Punktwolke an, um Verbindungen zwischen den Punkten herzustellen. Durch die Entwicklung von *Bewegungsprimitiven* auf den Quadriken wird der Roboter schliesslich auf der Oberfläche gesteuert.

Die Flächenabdeckungs- und Navigationslösungen wurden in Simulationen und Experimenten mit *e-puck* Robotern und *MagneBike* Robotern evaluiert. *MagneBike* Roboter sind kompakte Inspektionsroboter, die dank ihren magnetischen Rädern auf ferromagnetischen Strukturen klettern können. Im Hinblick auf die Mehr-Roboter-Inspektionsaufgabe wurde ein Bild basiertes Lokalisierungssystem bestehend aus einer *Kamera* und einem *Zielmodul* mit vier aktiven oder passiven visuellen Marken entwickelt und evaluiert. Es ist eine leichtgewichtige Lösung, um die vollständige *6D Position und Orientierung* eines Roboters zu lokalisieren. Sie ist besonders geeignet für die relative Lokalisierung von Inspektionsrobotern, die auf gekrümmten Oberflächen im 3D-Raum klettern können.

Die *Flächenabdeckungs- und Navigationslösungen*, die in dieser Dissertation entwickelt wurden, untersuchen Konzepte und bringen Konzepte ein, die von allgemeiner Bedeutung sind für die Umgebungsmodellierung, die Pfadplanung und die Koordination von Robotern in realen Umgebungen.

Stichworte: Mehr-Roboter-Systeme, Robotervertelung, Verteilte Algorithmen, Voronoi-Flächenabdeckung, Abdeckungspfadplanung, Pfadplanung in 3D-Umgebungen, Umgebungsmodellierung, LIDAR Punktwolken, Gitternetz-Erstellung, Gekrümmte Oberflächen, Fahrende und kletternde Roboter, Relative Roboterlokalisierung, Inspektion mit Robotern

Acronyms

1D	1-dimensional
2D	2-dimensional
2.5D	2.5-dimensional, 3D environment restricted to 2D plane and variable height in perpendicular direction
3D	3-dimensional
4D	4-dimensional
6D	6-dimensional
ACVT	Anisotropic Centroidal Voronoi Tessellation
CCVT	Constrained Centroidal Voronoi Tessellation
CGAL	Computational Geometry Algorithms Library
CVT	Centroidal Voronoi Tessellation
DCVT	Discrete Centroidal Voronoi Tessellation
DoF	Degrees of Freedom
ECM	Extended Marching Cubes mesh generation method
FOV	Field Of View
FTM	Fast triangulation mesh generation method
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IR	Infrared
ITM	Irregular triangular mesh generation method
LED	Light-Emitting Diode
LIDAR	Light detection and ranging device, or laser range finder

MAV	Micro Aerial Vehicle
MLS	Moving Least-Squares
NDT	Non-Destructive Testing
P3P	Perspective-3-Point
PCA	Principal Component Analysis
PCL	Point Cloud Library
PSM	Poisson surface mesh generation method
RBF	Radial Basis Function
RF	Radio Frequency
ROI	Region Of Interest
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneous Localization And Mapping
YUV	Color space with one brightness (Y) and two chrominance (UV) components

Contents

1	Introduction	1
1.1	Objectives	3
1.2	State-of-the-Art	7
1.3	Contributions	9
1.4	Organization	11
2	Environments and Robots	15
2.1	Environments	16
2.1.1	Steam Chest Environment	16
2.1.2	Test Setups	17
2.2	Robots	20
2.2.1	MagneBike Robots	20
2.2.2	e-puck Robots	23
2.3	Summary	23
3	Mathematical Methods	25
3.1	Notation and Terminology	25
3.2	Differential Geometry	28
3.2.1	Curved Surfaces	28
3.3	Computational Geometry	30
3.3.1	Voronoi Tessellations	30
3.3.2	Graph Search and Shortest Paths	35
3.4	Structure Inference	37
3.4.1	Basics of Tensor Voting	37
3.4.2	Tensor Voting Applied to Point Clouds	39
3.5	Control Theory	42
3.5.1	System Models	42
3.5.2	Gradient Descent Controller	45
3.5.3	Cooperative Control	46

3.6	Summary	52
4	The Concept of Hybrid Coverage	53
4.1	Related Work	54
4.2	Hybrid Coverage Solutions	56
4.3	Variants of Hybrid Coverage	57
4.3.1	Examples for Hybrid Coverage of Type 1	58
4.3.2	Examples for Hybrid Coverage of Type 2	61
4.3.3	Combination of Type 1 and Type 2	64
4.4	Results	64
4.4.1	Comparison of Hybrid Coverage Algorithms	65
4.4.2	Application of Hybrid Coverage to Inspection	70
4.5	Summary	72
5	Multi-Robot Coverage under Constraints	73
5.1	Related Work	74
5.2	Preliminaries	75
5.3	Voronoi Coverage in Nonconvex Environments	77
5.3.1	Problem Formulation	77
5.3.2	Gradient Projection Controller	79
5.3.3	Combining Voronoi Coverage with Path Planning	81
5.3.4	Properties of the Nonconvex Coverage Algorithm	86
5.4	Voronoi Coverage in Unknown Environments	88
5.4.1	DisCoverage and Star-Shaped Domains	88
5.5	Results	90
5.5.1	Evaluation of the Nonconvex Coverage Algorithm	90
5.6	Summary	96
6	Multi-Robot Coverage on Curved Surfaces	97
6.1	Related Work	98
6.2	Preliminaries	100
6.3	Voronoi Coverage on Curved Surfaces	101
6.3.1	Problem Formulation	101
6.3.2	Surface Coverage with Shortest Path Distance	104
6.3.3	Surface Coverage with Approximative Euclidean Distance	112
6.3.4	Properties of the Surface Coverage Algorithms	117
6.4	Extensions to Adaptive and Hybrid Coverage Control	120
6.4.1	Adaptive Surface Coverage	121
6.4.2	Application to Hybrid Coverage	122
6.5	Results	123

6.5.1	Comparison of Surface Coverage Algorithms	124
6.5.2	Extensions of Surface Coverage Algorithms	130
6.6	Summary	133
7	Mesh-Based Path Planning on Curved Surfaces	135
7.1	Related Work	136
7.2	Triangle Strip Planning	138
7.2.1	Problem Formulation	138
7.2.2	Environment Representation	138
7.2.3	Path Planner and Robot Control	142
7.2.4	Extensions	144
7.3	Results	145
7.3.1	Evaluation of Triangle Strip Planning	145
7.4	Summary	147
8	Point-Based Path Planning on Curved Surfaces	149
8.1	Related Work	151
8.2	Dense Point Cloud Planning	153
8.2.1	Problem Formulation	153
8.2.2	Environment Representation	154
8.2.3	Path Planner	157
8.2.4	Robot Control	164
8.3	Sparse Point Cloud Planning	167
8.4	Results	169
8.4.1	Evaluation of Tensor Voting	170
8.4.2	Evaluation of Dense Point Cloud Planning	173
8.5	Summary	181
9	Relative Robot Localization in 3D Space	183
9.1	Related Work	184
9.2	Preliminaries	186
9.3	System Overview	189
9.3.1	Camera and Target Modules	189
9.3.2	Optimization of Target Geometries	190
9.4	6D Relative Localization	192
9.4.1	Pose and Marker Prediction	193
9.4.2	Blob Extraction	193
9.4.3	Pose Update	195
9.5	Results	197
9.5.1	Localization of a Handheld Module	197
9.5.2	Aerial Vehicle Localization	199

9.5.3	Relative Localization for Multiple Robots	199
9.6	Summary	201
10	Conclusion	203
10.1	Discussion of Contributions	204
10.2	Outlook on Future Work	206
A	Proofs	209
A.1	P-Norms and the Parallel Axis Theorem	209
	Bibliography	213
	Curriculum Vitae	231

List of Figures

1.1	Industrial inspection.	2
1.2	Inspection sequence.	12
2.1	Steam chest environment.	17
2.2	Test setups.	18
2.3	Robot platforms.	21
2.4	Rotating LIDAR and 3D point clouds.	22
3.1	Tensor voting in three dimensions.	38
3.2	Tensor voting framework.	40
3.3	CVT and Voronoi coverage.	49
3.4	Distributed computation and gradient descent.	49
4.1	Concept of hybrid coverage.	57
4.2	Two types of hybrid coverage.	58
4.3	Combinations of hybrid coverage.	65
4.4	Coverage by flocking and sweeping.	66
4.5	Coverage by a robotic band.	66
4.6	Hybrid Voronoi coverage.	67
4.7	Comparison of hybrid coverage variants.	68
4.8	Expert knowledge and user guidance.	71
5.1	Transformation to star-shaped domain.	89
5.2	Voronoi coverage in a U-shaped environment.	91
5.3	Cost for covering the U-shaped environment.	91
5.4	Voronoi coverage in an environment with narrow passage.	92
5.5	Voronoi coverage in an environment with two free-standing obstacles.	93
5.6	Voronoi coverage in an environment with one nonconvex free-standing obstacle.	93

5.7	Voronoi coverage in a 3D nonconvex environment.	94
5.8	Voronoi coverage in an L-shaped environment.	95
5.9	Voronoi coverage in a U-shaped environment.	95
6.1	Graph-based representation.	101
6.2	Wavefront propagation.	111
6.3	Vertex exchange.	116
6.4	Hybrid surface area coverage.	123
6.5	Simulations on a curved surface.	125
6.6	Experiments on a curved surface.	127
6.7	Statistical evaluation.	128
6.8	Additional nonconvexity.	129
6.9	Adaptive surface coverage of a sphere.	131
6.10	Adaptive surface coverage of a torus.	131
6.11	Hybrid surface area coverage of a steam chest.	132
7.1	Triangle strip path and vector field generation.	143
7.2	Triangle strip path and vector field on Stanford Bunny.	146
7.3	Generated triangle meshes of industrial structures.	146
7.4	Simplified triangle mesh and planned triangle strip paths.	147
8.1	Token reduction of LIDAR point cloud.	155
8.2	Environmental feature maps.	156
8.3	Graph connectivity.	159
8.4	Discrete control set.	160
8.5	Robot control.	165
8.6	Paths on curved surfaces.	166
8.7	Sparse point cloud planning.	168
8.8	Robustness against variation in point density.	171
8.9	Variation of masking parameter β	172
8.10	Robustness against variation in noise.	173
8.11	Obstacle avoidance and negotiation.	175
8.12	Path planning on geometrically complex surface.	176
8.13	Path planning in the steam chest.	177
8.14	Computation times for navigation with dense point cloud planning.	178
8.15	MagneBike avoiding a hole obstacle.	179
8.16	MagneBike negotiating an edge obstacle.	180
9.1	The P3P problem.	187
9.2	Relative localization system.	191

9.3	6D relative localization.	194
9.4	Blob extraction.	195
9.5	Relative localization of a handheld module.	198
9.6	Relative localization of a flying quadrotor.	200
9.7	Relative localization of multiple robots.	201

Chapter 1

Introduction

An ever-growing infrastructure—including existing and newly built pipelines, piping systems, storage tanks and power plants—calls for periodic inspection and maintenance to guarantee its safe and efficient operation in the years to come. Many existing power plants are reaching the end of their designated lifespan. Aging, corrosion and mechanical stress cause material losses and structural damages, which may lead to leakages in the structures or their complete destruction (see Figure 1.1, bottom row). Inspection and maintenance allow for the early detection of defects and prevention of massive damages in installations, or of supply shortfalls. The periodic inspection of fossil and nuclear power plants is furthermore prescribed in safety and environmental regulations and is required by law.

However, the inspection task is often a difficult one, which is to this day conducted by human inspectors and involves environments with limited accessibility (see Figure 1.1, top row). *Non-destructive testing (NDT)* enables the inspection of industrial structures by reducing damage caused by the inspection to a minimum. Visual inspection, Eddy-current and ultrasonic testing are among the approved sensing methods commonly used for NDT. If a structure is not accessible disassembly is required, and parts need to be moved to a workshop for inspection. This involves increased risk of disassembly damage and high cost due to the outage of the facilities during the inspection procedure. If a structure can be accessed in the field, *in-situ* inspection with handheld devices, such as cameras or borescopes, is possible with the advantage of reducing the outage duration.

More recently, remote-controlled mobile systems have been increasingly considered for inspection. Such systems are semi-automated but typically clamped or mechanically guided, allowing movements with a few degrees of



Figure 1.1: Industrial inspection. Top row: NDT of industrial structures presents many challenges: manual work, need for reliability and expert knowledge, limited accessibility and safety issues. Bottom row: Typical ultrasonic NDT system with pulser/receiver, sensor probe and display device, scanning a metal test structure with buried defect; burst tube, which demonstrates possible consequences of missed defects at inspection. (Image sources: ALSTOM Power Services, Aqualified LLC, RIGCOM Access.)

freedom. The development of these systems is motivated by the search for more automated solutions and *assistive inspection tools* to address the major challenges of periodic inspection. Assistive inspection tools should provide for:

1. *Maximum reliability, safety and inspection quality.* Inspection tools provide sensing and actuation technology in order to achieve repeatable and persistent monitoring of a structure. Data recorded at periodic inspection runs is gathered in a consistent way over long time periods. An exact history of the measurements can be created and human errors reduced. Automated systems must be as reliable as traditional systems. Accurate localization and navigation systems are therefore key requirements of any automated solution.
2. *Maximum efficiency and reduction of outage time.* Most current inspection devices must be guided manually. These procedures require accurate professional skills and are time consuming. Automated inspection tools promise a speed-up through increased automation, par-

allelization of task execution and inspection during running operations. Societal dependence on power and on other basic supplies implies short outage durations. Moreover, reduced inspection time means reduced cost and increased profitability. Superior cost efficiency again enables more regular inspections.

3. *Maximum accessibility and coverage.* Certain parts under inspection are not reachable by human inspectors without risk or great difficulty. Even if a structure can be accessed with a sensor probe, the inspector's view might be occluded and an exact construction plan, such as a 2D drawing or 3D visualization, is usually not available. Inspection tools may be augmented by additional capabilities and offer aids in order to access otherwise inaccessible or unknown structures.
4. *Maximum flexibility and ease of use.* Industry lacks in experienced inspectors. Thus human experts should mainly be needed for analyzing the inspection results. In addition, manual inspection is demanding. Inspection tools can help to ease the process of inspection and make it scalable and applicable by non-experts. Industrial structures are often not standardized; therefore, inspection tools should furthermore adapt to moderately varying environments and conditions.

We believe that autonomous inspection robots can implement assistive inspection tools with a high potential to improve reliability, efficiency, coverage and flexibility of future inspection and maintenance.

1.1 Objectives

We draw our motivation from the inspection scenario. It presents an interesting and challenging problem for research in mobile robotics. The main objective of this thesis is to study multi-robot coverage and path planning methods with regard to autonomous coverage and navigation in environments with obstacles and curved surfaces. We analyze general theoretic concepts for covering complex geometries but also look at the more specific application of inspecting a steam chest environment with wheeled climbing robots.

We formulate our long-term vision and the concrete objectives of this thesis in the following. The remainder of this introduction is organized as follows. In Section 1.2, we summarize the state-of-the-art in related research areas of environment modeling, robot path planning and multi-robot coverage from the perspective of robotic inspection of curved surfaces and complex 3D

environments. The contributions of the thesis are stated in Section 1.3 and the thesis’s structure and content are outlined in Section 1.4.

Assistive Inspection Tools. Robotic inspection presents in its basic form a search, coverage or exploration problem. The *inspection task* is therefore composed of search, coverage and exploration tasks. If detecting the existence of a defect in a structure is sufficient, the problem of finding single defects can be viewed as a *search task*. In the case where a complete inspection is required, a complete search or *coverage task* is performed. Complete sensor coverage results in a map of the environment, which includes the locations of all detected defects. In addition, if the environment is unknown prior to inspection, an *exploration task* to explore the environment—either in a separate precedent exploration step or in a combined step of exploration and search or coverage—has to be conducted.

During the inspection task, an environment or surface is monitored from a distance or in close contact. A camera or laser sensor is moved through the environment or a NDT sensor of given footprint is swept in close proximity over the surface. We see the mobile robots as the *carriers* of such inspection tools; the robots carry an inspection sensor and move directly on the surface that needs to be inspected.

For practical realization, this requires a full mobile robot system. Appropriate sensors for the inspection as well as for the robots’ perception of the environment need to be incorporated. Locomotion and adhesion principles must allow for climbing surfaces and negotiating obstacles on the way. Global localization and mapping of complex 3D industrial structures must be provided, such that the robots can localize themselves, map the environment and report the locations of the detected defects accurately. The maps further enable 3D visualization, collision detection and navigation, including both local and global path planning on the surface. In addition, collaborative inspection with a group of robots needs for communication and coordination policies that orchestrate localization, environment modeling and planning among the robots.

Robot design, locomotion and adhesion, sensing and localization in the context of the inspection scenario have been the subject of two previous dissertations (Fischer, 2010; Tâche, 2010). In this thesis, we mainly address the problems of planning and multi-robot coordination.

Path Planning for Complex Environments. Our goal is to develop path planning algorithms that enable a robot to move in realistic 3D environments. The environments are composed of obstacles, uneven terrain with

varying navigability or the surfaces of 3D objects. Particularly the last type of environments presents a rather unique feature, which directly originates from climbing 3D structures, such as tanks and tubes of industrial plants. In these environments, climbing robots move along 6D trajectories through full 3D space but always remain constrained to the surface. The modeling of such environments presents a first challenge. As path planning is inseparably linked to the underlying environment representation, questions of how to represent the 3D environments and how to plan feasible and optimal paths through these environments both need to be answered. With respect to the motivated inspection scenario, we hope to provide navigation solutions that increase the autonomy of inspection robots and lead to inspection tools with improved repeatability, coverage and ease of use.

Toward Multi-Robot Coverage and Inspection. Based on the path planning capabilities of a single robot, our goal is to develop distributed coverage algorithms which deploy multiple robots in the environments. Again, the environments may include obstacles and can be curved. These constraints need to be included in the coverage solutions. Depending on the inspection task and sensor type, covering paths need to be generated, such that every location in the environment has been either scanned by a robot's sensor from a distance or visited by a robot for coverage with a contact sensor. With respect to our inspection scenario, we envision multiple inspection robots which parallelize the inspection process and exploit synergies, leading to increased inspection efficiency, robustness and quality.

Scope, Basic Assumptions and Limitations. Robotic inspection and multi-robot systems both represent an extensive branch of robotics research. As mentioned above, a single robot inspection task assumes a fully developed mobile robot system in place. Moreover, the inspection of industrial structures is particularly challenging due to the complexity of the environment. A multi-robot system then involves several such robots and focuses on their interaction. It becomes clear that, in order to make good progress, we need to narrow down the area under study and take meaningful assumptions.

Given the overall objective of increasing the autonomy for assistive inspection tools, and against the background that locomotion and localization have been addressed in previous dissertations, we focus on parts that have not yet been studied in the context of inspection. We extract some of the key challenges that occur along an inspection task, and contribute specifically, namely to environment modeling and path planning, multi-robot coverage and relative robot localization.

We make the following basic assumptions. The presented coverage solutions all build on the kinematic model of a single integrator. We assume that such a simple kinematic model is sufficient, since the high-level coordination policies can always direct a low-level controller to follow the velocity input.

The proposed coverage algorithms are designed to be decentralized and allow for distribution on multiple robot platforms, such that only wireless communication among neighboring robots in a limited communication range is required. However, our focus is not on communication between robot neighbors or between a robot and its base station as such. We do not study communication delays, losses of packets or signal strength, or interference and multi-path phenomena resulting from confined environments.

In terms of localization, we assume that a good estimate of the robot poses is provided by a localization module. If not otherwise stated, we assume ideal global localization of the robots. The localization in 3D space has been studied for the MagneBike robot platform by Tâche (2010) and Tâche et al. (2011), and an extension of the localization framework is presented by Pomerleau et al. (2011). Furthermore, experiments showed that, even though localization uncertainty is not modeled explicitly, the proposed coverage solutions are robust against modest localization errors.

We model the environments from 3D point clouds, which consist of a single laser scan or multiple scans that have been aligned beforehand. We do not consider dynamic map updates, incremental mapping and fusion of multiple robot maps. For the multi-robot coverage which involves mesh maps, we construct the underlying mesh first. In the cases where we assume that the robots do not know the mesh model, the mesh updates are simulated by robots that incrementally uncover the already constructed underlying mesh as they go. However, we have modeled environments from noisy data, and methods have recently been presented in the literature that allow for incremental dense reconstruction of the environment (e.g., Newcombe et al. (2011)), what makes us confident that our assumptions are not overly restrictive.

The main limitation is hence the lack of an underlying multi-robot simultaneous localization and mapping (SLAM) framework, which serves as basis for the higher-level navigation and coordination modules. Although most of the core components for a complete multi-robot inspection system are available, it requires a significant integration effort to combine localization, mapping, path planning and multi-robot coverage all together on an inspection robot platform like MagneBike. With respect to the inspection scenario, further open issues include the equipment of the robot platforms with NDT sensors and their test within the proposed navigation and coverage solutions, as well as the practical implementations of the robots' power supply and

communication during deployment inside the industrial structures.

1.2 State-of-the-Art

To start with, we provide an overview on different research activities that have addressed problems related to robotic inspection and navigation in 3D environments and environments with complex geometry; they have led to results relevant for building autonomous inspection tools. Additional related work, more focused on the respective problem, will be reviewed at the beginning of each chapter.

Mobile Robots for Uneven Terrain and 3D Environments. Robotic inspection systems often follow a clear trend: they include small-size climbing robots with high mobility but without much integration of sensing, localization and planning, or conversely, systems with advanced localization and mapping or planning capabilities but with limits in climbing or navigating 3D environments. The MagneBike robot, which is presented in Chapter 2 in more detail, is an attempt to combine both sides. Fischer (2010) and Tâche (2010) include overviews on locomotion principles for climbing robots, and Caprari et al. (2012) surveys robot systems designed for the inspection in power plants. Moser et al. (2007) presents an example for mobile inspection devices, which increasingly find use in industrial inspection. Alternatives to ground and climbing robots are aerial vehicles, which allow for high mobility in 3D environments and are becoming more and more popular. Burri et al. (2012) presents an example for their application in robotic inspection. Apart from inspection, robot systems for navigation in rough terrain and other difficult to navigate environments are advancing. Rusu et al. (2009) gives a representative example for an agile robot system with integrated localization, mapping and path planning. The special issue in the International Journal of Robotics Research on 3D Exploration, Mapping, and Surveillance (Michael et al., 2012) provides further insight in recent achievements of navigating robots in challenging 3D environments.

Perception and Modeling of 3D Environments. The aforementioned robot systems move on uneven terrain and in 3D environments. Methods that create 2.5D representations like elevation maps are commonly used to model rough terrain environments for navigation (Hebert et al., 1989; Thrun et al., 2004b). Extensions to multi-level surface maps (Triebel et al., 2006) introduce several surface classes per cell to allow for modeling vertical structures and environments with multiple overlapping levels, such as bridges or

underpasses. However, this solution is not intended for robots which traverse vertical surfaces and ceilings or move in full 3D space. Environments can alternatively be modeled by 3D evidence or voxel grids (Moravec, 1996; Wurm et al., 2010). Aside from grid-based representations, polygonal surface meshes are able to represent the surfaces of complex structures more accurately. However, 3D surface reconstruction methods undergo a permanent trade-off between computation cost and accuracy. Accurate but computationally demanding approaches from computer graphics (Kazhdan et al. (2006), and references therein) are confronted with simpler and faster but less accurate methods developed for robotic applications (Gingras et al., 2010; Marton et al., 2009). We will explore mesh-based representations further in Chapter 7. Another option is to work directly on the point clouds (Pomerleau et al., 2011; Smith et al., 2011; Vaskevicius et al., 2010; Vona and Kanoulas, 2011). This is beneficial when dealing with noise and topological distortions, since it makes the merging of data sets easier and avoids the cost of one entire processing step by bypassing explicit surface reconstruction. We will come back to point-based representations in Chapter 8.

Path Planning in 3D Environments. In general, search-based, sampling-based, as well as combinatorial planning all rely on a graph structure that connects through different states (LaValle, 2006). Graph search algorithms like Dijkstra’s algorithm, A* or D* algorithms (Dijkstra, 1959; Hart et al., 1968; Koenig and Likhachev, 2002; Stentz, 1995) are thus commonly used for path planning. Both the Theta* and the Field D* algorithms and their extensions to 3D (Carsten et al., 2006; Ferguson and Stentz, 2006; Nash et al., 2010), offer methods to generate paths that are no longer constrained to graph edges in 2D maps and 3D volumes. Kimmel and Sethian (1998) and Surazhsky et al. (2005) furthermore plan shortest paths on curved surfaces based on fast marching and window propagation methods, exploiting the surface mesh as graph structure. More specifically, Howard and Kelly (2007) and Wettergreen et al. (2010) plan trajectories for rovers over rough terrain; feasible trajectories are generated by forward simulation on mesh models. Wettergreen et al. (2010) uses A* search for global path planning. Howard and Kelly (2007) builds on a detailed vehicle model and an optimization-based trajectory generation framework with applications to local and global motion planning in the context of state lattices (Pivtoraiko and Kelly, 2005). Related to the application of robotic inspection of surfaces, the work by Englot and Hover (2011) deals with in-water ship hull inspection by an autonomous underwater vehicle. The ship hull is modeled as surface mesh from sonar range data and inspection paths encompassing the reconstructed model are

generated by sampling-based motion planning. However, the robot's motion resides in open 3D space and, different from our inspection scenario, is not directly constrained by the surface geometry.

Multi-Robot Coverage of Curved Surfaces and 3D Environments.

Atkar et al. (2005) presents a method to generate spray gun trajectories for the uniform coverage of topologically simple surface patches in automotive spray painting. A hierarchical automated segmentation procedure extends the approach to work for more complex surface geometries (Atkar et al., 2009). The idea of decomposing a surface area in order to cover simpler surface patches bears relations to our coverage solutions presented in Chapter 4 and Chapter 6. Extensions from single to multi-robot systems open up another research field. Correll and Martinoli (2009) presents a coverage algorithm for a miniature robot swarm which collaboratively inspects the compressor blades of jet turbines. A multi-robot system for ultrasonic NDT of industrial structures is developed by Hayward et al. (2006) and Dobie et al. (2007), with the main focus on the system's reconfigurability to realize a robust and adaptable distributed scanner. Tavakoli et al. (2012) uses a heterogeneous multi-robot system composed of a pole-climbing robot and multiple ground robots for automated inspection of the outer surface of industrial piping systems. The ground robots collaboratively map the structure to be climbed and localize the climbing robot. Due to the uniform geometry of the pipe structures, however, mapping and planning are largely reduced to 1D and 2D problems. Multi-robot coverage of 3D environments is alternatively studied for flying robots. Schwager et al. (2011) and Renzaglia et al. (2012), among others, deploy multiple aerial vehicles with down-looking cameras in 3D space to monitor an environment. The optimization-based coverage algorithms aim at maximizing the visible area while keeping mutual overlaps in coverage and distances between the vehicles balanced.

1.3 Contributions

In our work, we have developed mobile robots, and navigation and coverage solutions for their control. The presented algorithms apply in general to multi-robot coverage and path planning, and particularly under the constraints of a geometrically complex environment. Environments with obstacles and curved surfaces are characteristic for inspection scenarios. Our long-term goal is toward assistive inspection tools of increased autonomy, which enable the robotic inspection of infrastructures with benefits in inspection reliability and efficiency, accessibility and flexibility.

The contributions of this thesis are as follows.

1. *Hybrid Coverage Concept.* We present the concept of hybrid coverage, which suggests to combine deployment and sweeping motions in different scope and order. This offers an alternative view on existing coverage algorithms and proves helpful in developing new coverage algorithms.
2. *Coverage of Nonconvex Environments.* We combine Voronoi coverage with path planning. Original Voronoi coverage does not allow for obstacles and nonconvexities in the environment; the built-in path planner can resolve this issue. We further extend an exploration method related to Voronoi coverage to nonconvex environments. Our considerations focus on continuous space and include theoretical proofs. The coverage solutions are implemented and tested with a group of e-puck robots.
3. *Coverage of Curved Surfaces.* Our focus here is on coverage using a discrete representation of the environment. We use Voronoi coverage on a graph, which leads to solutions for 2D and 3D environments with obstacles and nonconvexities, and in particular for 2D manifolds embedded in 3D space. Ways of how to additionally control and adapt the resulting coverage are shown. Two coverage solutions are implemented, compared and tested with a group of e-puck robots.
4. *Mesh- and Point-Based Path Planning.* Environment modeling and path planning for 3D environments are inseparably linked. We present two navigation solutions for curved surfaces, which are based on two different representations of the environment. The mesh-based solution reconstructs a triangle mesh from a 3D point cloud and plans triangle strip paths on the mesh. The point-based solution augments a 3D point cloud and plans paths directly on the point cloud without further need of the mesh generation. Both path planners support the higher-level coverage solutions. They allow for robot navigation in realistic environments with obstacles and curved surfaces or rough terrain. The navigation solutions are implemented, and demonstrated in simulations and experiments with the MagneBike robot.
5. *6D Relative Localization System.* Relative localization between robots is a basic requirement for any multi-robot coordination. We support our coverage solutions with a lightweight vision-based relative localization system, which allows for 6D relative localization. It is particularly developed for the relative localization of climbing robots in 3D inspection tasks. The relative localization system is implemented, and demonstrated with a quadrotor helicopter and two e-puck robots.

1.4 Organization

First we show the organization of the thesis as it is motivated by the inspection scenario. Then we present the contents chapter by chapter.

Figure 1.2 gives a schematic overview of a typical sequence in robotic inspection. The key parts which we address throughout this thesis are highlighted. It starts with an industrial structure, which needs to be inspected. An inspection robot is launched. The robot uses its perception to sense the environment and record raw data, e.g., 3D point clouds from a laser range finder. The point cloud is preprocessed and features are extracted. The robot estimates its internal state by using odometry, and accelerometers or an IMU. Based on the estimated pose, the point clouds are registered and the robot is localized (Tâche, 2010). Either the environment is known, partially known or unknown. For unknown environments, a new map needs to be built. The map of the environment enables robot navigation. We present techniques for mesh reconstruction and augmentation of point clouds, as well as mesh-based and point-based path planning techniques in Chapter 7 and Chapter 8. The goal for the path planner is provided by user guidance or by a high-level coordination policy, which controls a mission or task, e.g., a coverage task. If multiple robots are involved, the robots localize each other, exchange information and agree on their current goals. Chapter 9 describes an example of a relative localization system, which can be used for inspection robots. Chapter 4, Chapter 5 and Chapter 6 present distributed coverage algorithms, or policies respectively, for the deployment of the robots and coverage of the structure. Local planners and low-level controllers move the robots along the planned paths toward a specified goal. Obstacles are detected and are either avoided or negotiated. Principles of robot locomotion and obstacle negotiation are studied by Fischer (2010).

The inspection sequence actuates the inspection robot and, when repeated, moves it through the structure. If a location is reached that needs to be inspected, the robot can activate carried inspection sensors to scan the surrounding surface.

The thesis begins with an introductory part about robots, environments and mathematical methods used, which forms the basis for the later chapters. We discuss different multi-robot coverage concepts and transition to the main chapters of the thesis, which present the contributions on multi-robot coverage and path planning. Each chapter has a short introduction, which motivates the chapter, a related work section, which presents further work that is specifically related to the chapter, and a main section, which presents

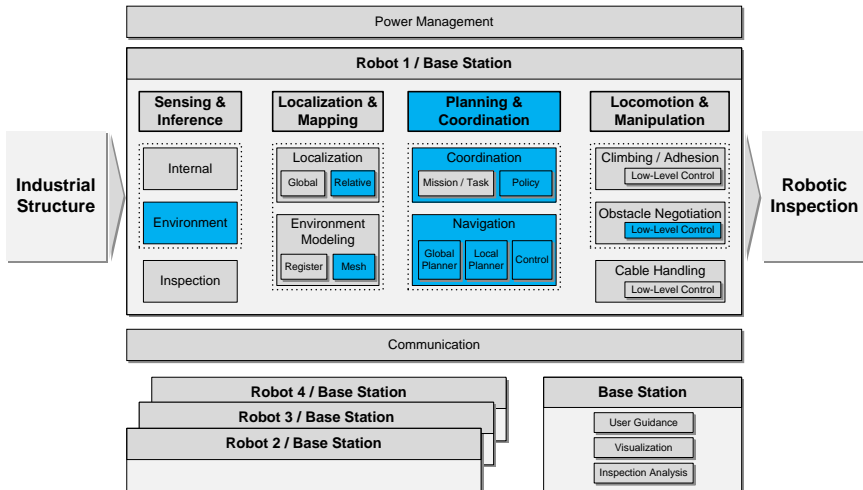


Figure 1.2: Inspection sequence. In order to achieve robotic inspection of an industrial structure, a robot needs modules for sensing, localization and mapping, path planning and coordination, locomotion and obstacle negotiation, communication and power management. The modules that are mainly covered by our work are highlighted in blue. The coordination of multiple robots through coverage policies is presented in Chapter 4, Chapter 5 and Chapter 6. Environment modeling and robot navigation are the topics of Chapter 7 and Chapter 8. Relative robot localization is finally addressed in Chapter 9.

the methods and results. Each section concludes with a summary of the chapter, including a valuation of the main achievements.

Chapter 2 introduces the environments that were used for simulations and experiments, as well as a typical industrial structure, which represents a case study and benchmark for the robotic inspection. Furthermore, the used robot platforms, the MagneBike and e-puck robots, are presented.

Chapter 3 gives definitions and describes mathematical methods to provide a theoretical background for the later developments. It covers necessary basics from differential geometry, computational geometry, structure inference with tensor voting, robot path planning as well as feedback and cooperative control. In particular, the important concepts of Voronoi tessellations and Voronoi coverage are introduced.

Chapter 4 explains the hybrid coverage concept. Several examples of hybrid coverage variants are provided. Some variants are evaluated in simulation and the hybrid Voronoi coverage method is found to be particularly promising, especially with respect to the inspection scenario.

Chapter 5 addresses the extension of Voronoi coverage to handle obstacles and nonconvexities. The nonconvex coverage algorithm formulates a constrained optimization problem and uses ideas from the gradient projection method from optimization. Voronoi coverage is combined with robot path planning to find a solution. Besides, the transformation for extending the DisCoverage exploration algorithm to environments with arbitrary obstacles is described.

Chapter 6 applies the Voronoi coverage method to curved surfaces. We use triangle meshes to represent the environments. Two adaptive coverage algorithms are presented, which both deploy multiple robots into discrete Voronoi partitions over the surfaces. In order to generate a Voronoi partition, the first coverage solution propagates a discrete wavefront over the mesh, whereas the second solution exchanges vertices of the mesh locally between robots.

Chapter 7 describes meshing and path planning methods for mesh-based path planning. Several state-of-the-art surface reconstruction methods are evaluated and triangle strip paths are planned over realistic 3D meshes. The robots are guided by a continuous vector field controller. In support of the surface coverage and inspection, covering triangle strips are constructed to completely cover the surface area.

Chapter 8 presents the alternative point-based path planning. 3D point clouds of curved surfaces are preprocessed and augmented by tensor voting. Using dense point cloud planning, a path is planned over the surface, taking noise in the point cloud, local surface characteristics and obstacles on the surface into account. Sparse point cloud planning represents an alternative approach, where geometric shapes are fitted to a sparser point cloud and motion primitives are developed on the fits.

Chapter 9 gives a description of a 6D relative localization system for relative localization of multiple robots in a 3D environment. The vision-based localization system is compact and lightweight and consists of two complementary modules. The target module is detected and tracked by a camera module, which allows for the estimation of the target's pose.

Chapter 10 concludes the thesis and gives an outlook on future work.

Chapter 2

Environments and Robots

This chapter presents the environments and robot platforms used in this thesis. We describe a characteristic environment for the robotic inspection task, as well as the different environments and test setups we used for the simulations and experiments. Furthermore, the *MagneBike* robots and the *e-puck* robots are introduced.

Our intention in this chapter is threefold. First, the chapter complements the motivation of Chapter 1 by giving concrete examples for an industrial environment that needs to be inspected and a group of climbing robots that executes the inspection task. An overview of the key figures of the steam chest environment and the *MagneBike* robots is provided. Second, equipping the reader with knowledge of a typical environment and robot platform with respect to the inspection scenario helps in understanding decisions that are motivated by the real application. The environment characteristics and the robot design serve as a starting point for the development of many of the concepts and algorithms in this thesis. Third, the chapter presents the environments and robots that are used in simulations and experiments throughout the thesis. The specifications of the test setups and robots contain useful information, which supplement the discussions in the results sections of the following chapters.

The specifications of the steam chest environment and the *MagneBike* robot have previously been reported in parts in the dissertations of Fischer (2010) and Tâche (2010), and at conferences (Breitenmoser et al., 2010c) and in journals (Stumm et al., 2012; Tâche et al., 2009, 2011). The chapter is divided into two main sections. Section 2.1 describes the different environments and Section 2.2 presents the *MagneBike* and the *e-puck* robots. The chapter is concluded with a summary in Section 2.3.

2.1 Environments

We first present an example of a typical steam chest environment. Some specific challenges and characteristics of the environment are highlighted, which provide the specifications for the inspection task. Moreover, we describe the test setups built for the experimental evaluation of the developed navigation and coverage solutions.

2.1.1 Steam Chest Environment

Steam chests are ferromagnetic tube-like supply structures for feeding hot steam into the steam turbines of a power plant (see Figure 2.1, left). The high temperature and pressure of the steam as well as vibrations from the turbines cause stress in the structure and lead to defects or degradation of the material. In order to inspect steam chests by conventional inspection tools, disassembly of the structure is necessary. Handheld borescopes cannot be used because of the high number of bends or intersections and the large diameter of the pipes.

Steam chests represent a challenging environment for mobile robotics. They are three-dimensional, and their geometry is complex and constraints arise from different types of obstacles, narrow sections and sections of varying sizes. A climbing inspection robot must be able to bring a sensor to any point on the curved surface, independently of the robot’s orientation with respect to gravity. Another challenge is the lack of natural light and perceptual features. The metal walls may be reflective and the texture is mostly uniform. Artificial landmarks or external beacons cannot be installed as accessibility or connectivity is limited by the closed metal structure. Sensors that may operate under such conditions need to meet further requirements of desired sensor range and accuracy, limited power consumption, size and payload. Furthermore, exact 3D models of the steam chests are generally not available at the time of inspection, since plans are outdated or simply not existent. Methods for environment modeling must allow for the representation of the topology of the closed surfaces; the environment is beyond general 2.5D and localization of a robot’s full 6D pose is required. A comprehensive list of requirements for an inspection robot with respect to locomotion and localization in a steam chest environment is provided by Tâche (2010).

The steam chest of Figure 2.1 was put out of service and is used as a benchmark in our work. It is about 5 m long, has seven entry points and the pipe diameters vary between 20 cm and 70 cm. Figure 2.1 on the right shows a 3D point cloud of the steam chest, which was collected in the course

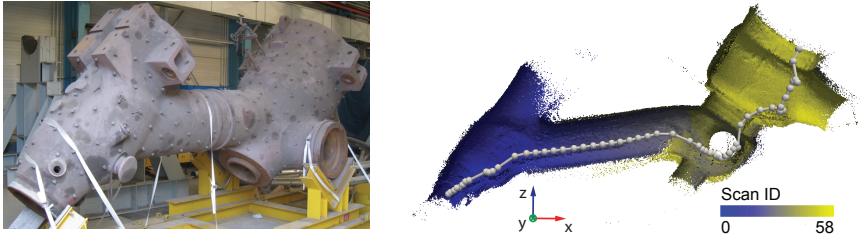


Figure 2.1: Steam chest environment. Left: Example of the steam chest which was used as benchmark in our work. Right: Cut view of the registered 3D point cloud (the shading of the points follows the scan acquisition numbers, the light gray line represents the trajectory of the rotating Hokuyo URG-04LX laser range finder, and the balls show the scanner poses where MagneBike took a scan).

of a field experiment, where the MagneBike robot was navigated through the environment by remote control. The robot was localized by the localization method presented by Tâche et al. (2011). For the point cloud, 59 scans were recorded, using the rotating Hokuyo URG-04LX laser range finder mounted on the MagneBike robot, and registered along a 5.8 m long path. This corresponds to roughly one scan registered every 10 cm along the path.

2.1.2 Test Setups

Next we describe the test setups which were used in the experiments for the coverage solutions of Chapter 5 and Chapter 6, and the navigation solution of Chapter 8.

Steam Chest Mock-Up

For realistic evaluation of the developed methods, a test setup was built to mimic the expected steam chest environment (see Figure 2.2, left). The test setup consists of a steel pipe composed of three parts: one main cylinder of 0.8 m diameter and 1.9 m length, an expanded section of 1 m diameter and 0.5 m length, and a small cylinder, 0.4 m in diameter and 0.6 m in length, branching off from the main section. We use a Vicon motion capture system¹ to localize the robots during the experiments and to provide ground truth of the robots' trajectories. The top portion of the main pipe can be taken

¹<http://www.vicon.com>

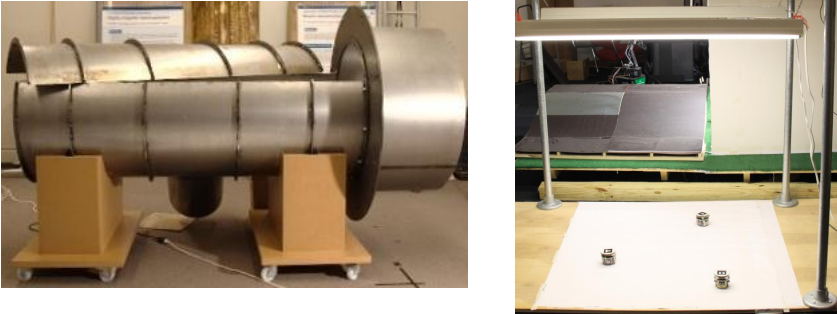


Figure 2.2: Test setups. Left: Steam chest mock-up with sections of different diameters and lengths. Right: Test setup with overhead camera, which tracks the three e-puck robots deployed on the planar ground plate. The bumpy slope test setup covered with dark brown flexible magnets can be seen in the background.

off in order to allow the cameras of the Vicon motion capture system to see inside and track the robots. This test setup was used for the experiments of Chapter 8.

Planar and Bumpy Slope Test Setups

We built a test setup with an overhead camera to track a group of robots (see Figure 2.2, right). We used cameras with resolutions of 1280×960 and 1280×1024 , respectively. The obtained localization accuracy is in the range of 1 cm up to 3 cm. The camera captures the images and passes them on to a base station. The images are read into the ARToolKit (Kato and Billinghurst, 1999), which extracts the markers carried by the robots and determines their pose. The navigation and coverage solutions are then executed on the base station to compute the robots' action for the next time step. The base station continuously sends information over Bluetooth and operates the robots by remote control. Each robot receives the commands and actuates its stepper motors, which closes the control loop.

The low-level control runs on the robots. The algorithms for tracking and for the actual path planning and coordination, running on the base station, use *Matlab*². This way, we can reuse the code base from the Matlab simulations, which provides us with high flexibility in respect of rapid-prototyping

²<http://www.mathworks.ch>

and testing. The drawback lies in Matlab's relatively low speed. Solutions are provided by Matlab's MEX files or real-time toolboxes.

The test setup comes in two versions. The first version consists of a planar $1\text{ m} \times 1\text{ m}$ ground plate. The camera is mounted in a distance of 1.35 m above ground. This setup was used for the experiments of Chapter 5; a similar test setup was used in the experiments of Chapter 9. The second version is a bumpy slope of $1.2\text{ m} \times 1.2\text{ m}$ base area. The curved surface is built from plywood and a thin carbon steel sheet covered with a flexible magnet is laid on top of the wooden frame. This version of the test setup was used for the experiments of Chapter 6.

Virtual Environments

We implemented the algorithms for the navigation and coverage solutions in Matlab and C++. Matlab and the *robot operating system (ROS)*³ are used as simulation environments. ROS wrappers are provided for some of the C++ implementations. The C++ implementations interface the point matcher library (Pomerleau et al., 2011), and depend on the external libraries *Eigen*⁴ for linear algebra, *Nabo*⁵ for nearest neighbor search and *OpenMesh*⁶ for mesh representation. OpenMesh provides an efficient halfedge-based data structure for representing and operating on triangle meshes. The system furthermore offers optional interfaces to the powerful computational geometry, point cloud and mesh processing libraries *CGAL*⁷ and *PCL*⁸.

Various standard 3D models from computer graphics, CAD models of the bumpy slope test setup, as well as real 3D point clouds recorded with the rotating Hokuyo URG-04LX and Hokuyo UTM-30LX laser range finders (see also Figure 2.4) were used for the simulation experiments.

³Robot Operating System, <http://www.ros.org>

⁴Eigen library, <http://eigen.tuxfamily.org>

⁵Nabo library, <http://github.com/ethz-asl/libnabo>

⁶OpenMesh library, <http://www.openmesh.org>

⁷Computational Geometry Algorithms Library, <http://www.cgal.org>

⁸Point Cloud Library, <http://pointclouds.org>

2.2 Robots

In the following, we present the two robot platforms we have mainly worked with.

2.2.1 MagneBike Robots

A strong motivation for this thesis originates from the development of the MagneBike robots (see Figure 2.3, left) and their application, the inspection of industrial environments, such as the steam chest environment. The main focus of this thesis is on the study of the methods for autonomous navigation and multi-robot coverage. However, during the last years another challenge was the ongoing development of the MagneBike robots. We will take this opportunity to present a brief summary of the MagneBike robots; a more detailed description of the robot platform can be found in the works of Tâche et al. (2009), Tâche (2010), Fischer (2010), and Caprari et al. (2012). The MagneBike robot exists in an older prototype version and a newer revised version. Five units of the newer version of the MagneBike robot have been prepared, and three of them are currently assembled, programmed, and tested under laboratory conditions.

MagneBike is a compact and lightweight climbing robot ($18.5 \text{ cm} \times 14.3 \text{ cm} \times 17.0 \text{ cm}$, 3.3 kg), which consists of two magnetic wheel units in a motorbike arrangement. The wheels are enhanced with lateral lifters, or respectively, with passive parallel wheels in the newer version of the robot's wheel unit (Fischer, 2010), which stabilize the robot and enable the robot to negotiate step-like obstacles. MagneBike's locomotion and adhesion concepts with an active degree of freedom on the front steering wheel and permanent magnets integrated into both wheels allow driving on ferromagnetic surfaces of industrial structures, which were not designed specifically for robots. The robot can climb vertical walls, follow circumferential paths inside pipe structures and pass over complex combinations of convex and concave step obstacles with almost any inclination regarding gravity. It requires only limited space to maneuver because turning on spot around the rear wheel is possible.

MagneBike is integrated with electronics on-board; it features up to five Maxon motors and encoders for actuation of the two wheels, the steering fork, and the two pairs of lateral lifters. Further sensors of the robot are a three-axis accelerometer, and strain gauges in the lifters and the robot body to measure forces and deformations. The platform is composed of a Gumstix Verdex Pro single-board computer running Linux. ROS is used as the overall

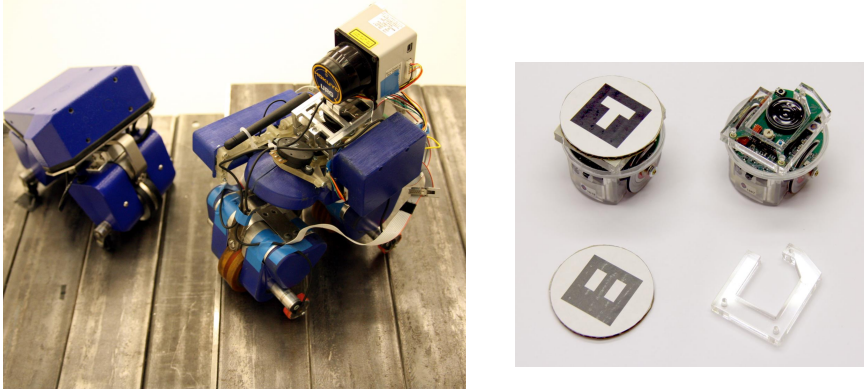


Figure 2.3: Robot platforms. Left: Two MagneBike robots, one equipped with the rotating Hokuyo URG-04LX laser range finder. The newer revised version (left) and older prototype version (right) are shown. Right: Two e-puck robots, equipped with visual markers for tracking by an overhead camera.

framework and is cross compiled for the ARM architecture of the Gumstix (GumROS). The low-level components are executed as ROS nodes directly on the robot, including low-level control, actuation of motors and readout of sensors. Other higher-level ROS nodes, such as the modules for environment modeling, path planning, multi-robot coordination or visualization and graphical user interfaces, run on a base station. The connection to the robot is established either via cable and Ethernet or over a wireless network. The robot is currently tethered and connected to an external power supply of 24 V. MagneBike can optionally be equipped with a rotating LIDAR for recording 3D point clouds of its environment (see Figure 2.3 and Figure 2.4, left). The scanner module consists of a Hokuyo URG-04LX laser range finder in the case of the older prototype, and for the newer platforms, a rotating laser range finder based on the Hokuyo UTM-30LX is currently developed⁹.

The recorded 3D point clouds are usually noisy, anisotropic and show high variations in density. In addition, specular reflections occur on metal surfaces, which results in increased deformations of the point clouds. These reflections do not only depend on the surface itself, but also change with the type and characteristics of the laser range finders used. Figure 2.4 on the right shows 3D point clouds obtained with the Hokuyo URG-04LX and Hokuyo UTM-30LX laser range finders. From our experience, the UTM-

⁹<http://www.hokuyo-aut.jp>

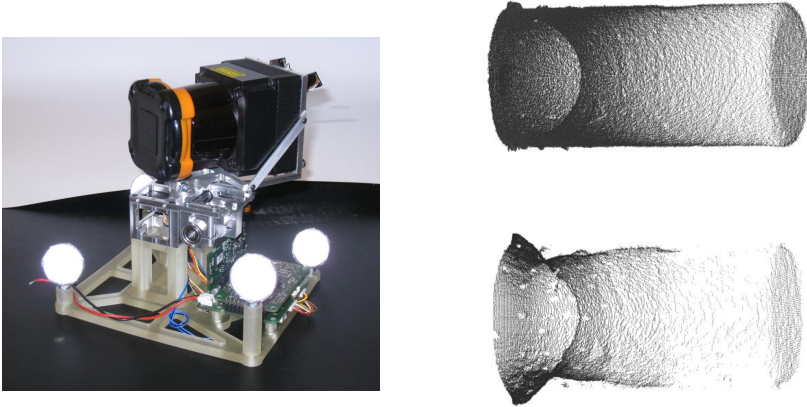


Figure 2.4: Rotating LIDAR and 3D point clouds. Left: Rotating Hokuyo UTM-30LX laser range finder; the scanner is developed as environment sensor for the newer MagneBike versions but is, as shown, also designed for stand-alone use. Right: 3D point clouds of a tube-like structure, obtained with the Hokuyo UTM-30LX (top) and Hokuyo URG-04LX (bottom) laser range finders. Note how the sensor resolutions, sensitivity to reflections and uniformity of the recorded point clouds vary for the two laser range finders.

30LX laser range finder provides significantly better results in both range and accuracy, especially in reflective environments. For a detailed analysis of LIDAR performance and its change for different range sensors and different surfaces in industrial environments, refer to our work presented by Pomerleau et al. (2012).

The 3D point clouds obtained from the laser range finders can be used to perform 6D global localization of the MagneBike robot (Tâche, 2010; Tâche et al., 2011). The localization is a sequential process. The robot uses readings from the wheel encoders and orientation information from the three-axis accelerometer to compute an estimate of its current pose, what is called *3D odometry*. After a certain distance has been traveled, the robot stops and takes another full 3D scan of the environment. The 3D transformation between the two last 3D scans is computed by the iterative closest point (ICP) algorithm (Besl and McKay, 1992), using the 3D odometry estimate for initialization. While the robot navigates, this process of local scan registration is repeated; robot position and orientation are updated, and a representation of the environment from aligned point clouds is incrementally constructed (see

Figure 2.1, right). In this thesis, we will use such point cloud representations as input to our navigation solutions.

2.2.2 e-puck Robots

For our multi-robot experiments, we use several e-puck robots (Mondada et al., 2009). The e-puck robot is a small two-wheel differential drive robot with a diameter of 7 cm (see Figure 2.3, right). e-puck is equipped with a dsPIC microcontroller, various sensors (e.g., IR proximity sensors) and actuators (e.g., LEDs). It is powered by a Li-ion battery, and offers a RS-232 and a Bluetooth interface for communication.

We kept the kinematic model of the robots in the experiments simple: the e-puck robots rotate until their front direction is aligned nearly parallel to the next goal direction (typically within $\pm 5^\circ$) and then drive straight to the goal.

We additionally equipped the e-puck robots with visual markers on the top for tracking by an overhead camera, and small magnets at the bottom for climbing ferromagnetic surfaces like the bumpy slope test setup.

2.3 Summary

This chapter presents the relevant environments and robots used in the context of this thesis. The steam chest environment represents a typical industrial structure which could be inspected by climbing robots like MagneBike. The MagneBike robot platform is a compact wheeled climbing robot designed for inspecting curved surfaces of complex 3D structures. Localization, environment modeling, path planning and multi-robot coordination for the MagneBike robots and their unique environments pose challenging problems, which are of general interest for mobile robotics.

In addition, we describe the e-puck robot platform and the test setups used for the simulations and experiments presented in this thesis. We will come back to these environments and robots repeatedly throughout the sections of the following chapters.

Chapter 3

Mathematical Methods

This chapter introduces the theoretical background for the work of this thesis. It covers basics of differential geometry in Section 3.2 and computational geometry in Section 3.3, describes a specific computational framework for structure inference in Section 3.4 and summarizes concepts from control theory in Section 3.5. First, we define some relevant notation and terminology in Section 3.1.

3.1 Notation and Terminology

General Notations

We let \mathbb{N} be the set of all natural numbers including zero, and define the set of positive natural numbers as $\mathbb{N}_{>0} = \mathbb{N} \setminus \{0\}$. $\mathbb{R}_{\geq 0}$ denotes the set of nonnegative real numbers. $N \in \mathbb{N}_{>0}$ is the number of dimensions, e.g., \mathbb{R}^N is the Cartesian product $\mathbb{R} \times \mathbb{R}^{N-1}$ and defines a N -dimensional vector space over the real numbers.

A subset $\mathcal{X} \subset \mathbb{R}^N$ is called *convex* if and only if, for any pair $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and $t \in [0, 1]$,

$$t \mathbf{x} + (1 - t) \mathbf{y} \in \mathcal{X}, \quad (3.1)$$

i.e., all points along the line segment that connects \mathbf{x} and \mathbf{y} are contained in \mathcal{X} . A *nonconvex* set is now defined as a set that is not convex.

A subset $\mathcal{X} \subset \mathbb{R}^N$ is called *star-shaped* if and only if there exists a point $\mathbf{x} \in \mathcal{X}$, such that, for all $\mathbf{y} \in \mathcal{X}$, the line segment from \mathbf{x} to \mathbf{y} is contained in \mathcal{X} .

A *metric* is a distance function which defines a distance between two elements of a set. An important class of metrics over \mathbb{R}^N is induced by the

p -norm, which is given by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^N |x_i|^p \right)^{1/p}, \quad (3.2)$$

with $p \in \mathbb{N}_{>0}$ and $i \in \mathbb{N}_{>0}$. If not otherwise stated, we assume in the following that indices i, j, k, l or similar are elements of $\mathbb{N}_{>0}$.

For $p = 2$ we get $\|\mathbf{x}\|_2$, which is the common *Euclidean norm*. The Euclidean norm induces the *Euclidean metric* or *Euclidean distance*. The *Euclidean space* is the space of *Euclidean geometry*, i.e., it is the normed vector space with the *Euclidean inner product* or *dot product* defined on \mathbb{R}^N . Euclidean spaces are distinguished from *curved spaces* of *non-Euclidean geometry*, where all axioms of Euclid are satisfied except for the parallel axiom (Kühnel, 2006). Refer also to Section 3.2 below.

According to our convention, vectors and matrices are written in bold type, \mathbf{x} and \mathbf{R} . If not otherwise stated, the vectors are assumed to be column vectors. Row vectors are given by $\mathbf{x}^T = [x_1 \dots x_i \dots x_N]$ for N dimensions. Vectors that are composed of n column vectors of dimension N are written as column vector $\mathbf{X} = [\mathbf{x}_i]_{i=1}^n = [\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n]$, with each $\mathbf{x}_i \in \mathbb{R}^N$ and $\mathbf{X} \in \mathbb{R}^{nN}$. Furthermore, we write in abbreviated form $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$.

The translation vector to express frame A (attached to object obj) with respect to frame B is denoted by \mathbf{t}_{BA} or $\mathbf{t}_{\text{obj},B}$, and a rotation matrix to express a vector given in frame A with respect to frame B is given by \mathbf{R}_{BA} . The transformation of combined translation and rotation is written as $[\mathbf{R}_{BA} \mid \mathbf{t}_{\text{obj},B}]$.

The “ $\hat{}$ ” symbol represents unit vectors or normalized vectors, $\hat{\mathbf{x}}$. The “ $\bar{}$ ” symbol denotes the mathematical average or mean, $\bar{\mathbf{x}} = 1/n \sum_{i=1}^n \mathbf{x}_i$. The prime “ $'$ ” means a vector that is projected, constrained or restricted to a set, \mathbf{x}' , and the “ $\tilde{}$ ” symbol denotes an estimated or modified quantity, $\tilde{\mathbf{x}}$.

For the function $f: \mathbb{R}^N \rightarrow \mathbb{R}$ and vector $\mathbf{x} \in \mathcal{X}$, the gradient is defined as the N -dimensional row vector $\nabla f(\mathbf{x}) = [\partial f / \partial x_1 \dots \partial f / \partial x_i \dots \partial f / \partial x_N]$. The time derivative is given as $\partial \mathbf{x}(t) / \partial t = \dot{\mathbf{x}}(t) = [\dot{x}_1(t), \dots, \dot{x}_i(t), \dots, \dot{x}_N(t)]$.

We write an objective function or cost \mathcal{H} in one of two alternative forms. $\mathcal{H}(\mathcal{X})$ denotes the function $\mathcal{H}: \mathbb{R}^N \rightarrow \mathbb{R}$, and $\mathcal{H}(\mathbf{X})$ is $\mathcal{H}: \mathbb{R}^{nN} \rightarrow \mathbb{R}$. A *critical point* or *optimizer* of the objective function \mathcal{H} is denoted by the “ $*$ ” symbol, \mathbf{x}^* and \mathbf{X}^* respectively, which are solutions of $\nabla \mathcal{H} = \mathbf{0}^T$.

A graph $G = \{V, E\}$ is composed of a finite set of vertices, V , and an edge set $E = V \times V$, with edges $e = (v, w) \in E$, $v \neq w$, and $v, w \in V$. The n vertices v_i of a graph are points of a discrete set, and we write for a graph embedding in \mathcal{X} , $\mathbf{v}_i \in \mathcal{X}_G$ and $\mathcal{X}_G \subset \mathcal{X}$, or $\mathbf{X}_G = [\mathbf{v}_i]_{i=1}^n$, where \mathbf{v}_i denotes the vector that describes the position of the graph vertex v_i in \mathcal{X} .

An objective function \mathcal{H}_G can then be defined accordingly for the discrete set of points \mathcal{X}_G or \mathbf{X}_G , respectively. A *one ring neighborhood* $\mathcal{N}_i^{\text{ring}}$ of a vertex v_i is the set of $m = |\mathcal{N}_i^{\text{ring}}|$ vertices $v_j = v_{n_{il}}$, $l \in \{1, \dots, m\}$, that are connected to v_i via an edge $e_{ij} = (v_i, v_{n_{il}})$.

Robot-Specific Terminology

Our understanding of a *robot* or *robot system* in this thesis encompasses systems that are decision-making and are provided by sensing and actuation devices for perception, communication, reasoning, planning and execution. If not otherwise stated, we assume that each robot r_i of a group of n homogeneous robots is equipped with several modules, including sensing, localization, environment modeling, communication, path planning, and high-level coordination modules. An environment sensor (e.g., a LIDAR) allows to record data from the environment within a sensor range of radius R_{sens} for environment modeling and localization, and an inspection sensor (e.g., a NDT probe) allows to detect points of interest, such as defects, in the robot's environment. A communication device enables the exchange of information among neighboring robots within a communication range of radius R_{com} . Two robots are said to be neighbors if they are within communication range.

The world or environment which the robots operate in is referred to as the *workspace* $\mathcal{W} \subset \mathbb{R}^N$ of the robots. A workspace includes besides the region of the n robots, $\mathcal{R} = \{r_i\}_{i=1}^n \subset \mathcal{W}$, the free space $\Omega \subset \mathcal{W}$ and the obstacle space $\mathcal{O} \subset \mathcal{W}$. Other robots in the workspace need to be avoided. Obstacles may be negotiated with some effort, i.e., they are part of Ω , or cannot be passed and need to be circumvented by the robots, i.e., they are part of \mathcal{O} . $\partial\Omega$ denotes the boundary of Ω and $I(\Omega) = \Omega \setminus \partial\Omega$ represents the interior of Ω . The positions of the n robots r_i are given by the set $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^n$, or vector $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n \in \Omega^n$ alternatively, with $\mathbf{p}_i \in \Omega$ representing the position of a single robot. The robot position on a graph G , $\mathbf{v}_{\mathbf{p}_i} = \mathbf{p}'_i$, results from the robot's position \mathbf{p}_i constrained onto the graph. In addition, a robot might have an orientation; the 3D position and the 3D orientation of a robot in a 3D environment define the 6D robot pose.

For path planning, yet another space, the *configuration space* \mathcal{C} , or more generally, the robot's *state space* \mathcal{X} , is of great importance. The configuration space for a robot with m -DoF, or the state space with m components respectively, is a topological m -dimensional manifold and encompasses the set of all rigid-body transformations or state transitions which can be applied to a *robot configuration* or *state* \mathbf{x} (LaValle, 2006). The *control inputs* or *actions* \mathbf{u} , which trigger such configuration and state transitions, are contained in a predetermined *control* or *action space* \mathcal{U} .

3.2 Differential Geometry

This section introduces some of the necessary tools to describe curved surfaces and shortest paths on the surfaces in mathematical terms. We closely follow the book by Kühnel (2006).

3.2.1 Curved Surfaces

A *surface* in \mathbb{R}^3 can generally be described for the components (x, y, z) by a function $F: \mathbb{R}^3 \rightarrow \mathbb{R}$ as $F(x, y, z) = 0$, whenever it holds $\nabla F \neq \mathbf{0}^T$. Examples are *planes* or *quadric surfaces*. A quadric surface or quadric can be defined by $\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{P}^T \mathbf{x} + C$, with $\mathbf{x} \in \mathbb{R}^3$, and for a 3×3 matrix \mathbf{Q} , a vector $\mathbf{P} \in \mathbb{R}^3$ and a constant C .

The *surface integral* over the surface area $A \subset \mathbb{R}^3$ with real-valued function $g(\mathbf{x})$ is written as

$$\int_A g(\mathbf{x}) dF(\mathbf{x}), \quad (3.3)$$

where $dF(\mathbf{x})$ denotes a *surface element*.

Manifolds

A *m-dimensional submanifold* $\mathcal{M} \subset \mathbb{R}^N$ is defined by the *local parametrization* $f: \mathcal{U} \rightarrow \mathcal{M}$, where $\mathcal{U} \subset \mathbb{R}^m$. m is the *dimension* and $N - m$ denotes the *codimension* of the submanifold \mathcal{M} . \mathbb{R}^N is the *ambient space*, and $\phi := f^{-1}$ is called a *chart* of \mathcal{M} .

$T\mathbb{R}^N := \mathbb{R}^N \times \mathbb{R}^N$ is the *tangent bundle* of \mathbb{R}^N . For every fixed point $\mathbf{x} \in \mathbb{R}^N$, the space $T_{\mathbf{x}}\mathbb{R}^N := \mathbf{x} \times \mathbb{R}^N$ denotes the *tangent space* at the point \mathbf{x} . The tangent space of the m -dimensional submanifold \mathcal{M} at the point $\mathbf{q} \in \mathcal{M}$ is the subspace $T_{\mathbf{q}}\mathcal{M} \subset T_{\mathbf{q}}\mathbb{R}^N$. The tangent bundle of \mathcal{M} then is the collection of tangent spaces, $T\mathcal{M} := \bigcup_{\mathbf{q} \in \mathcal{M}} T_{\mathbf{q}}\mathcal{M}$.

\mathcal{M} is further called *orientable* if there exists a definite choice of rotational directions in each tangent plane, which is not changed within the individual charts.

Intuitively speaking, a *m-dimensional differentiable manifold* \mathcal{M} is a topological space that in the vicinity of every point \mathbf{q} resembles \mathbb{R}^m , i.e., *locally* or *in the small* the topology of a manifold is the same as that of \mathbb{R}^m . In particular, every m -dimensional submanifold $\mathcal{M} \subset \mathbb{R}^N$ is also a m -dimensional manifold. However, a manifold can be described in terms of local coordinates in the form of parametrizations or charts, such that it does not necessarily rely on an ambient space.

Curved spaces are typically described by *Riemannian geometry*. The central subject of Riemannian geometry is the *Riemannian manifold*. A Riemannian manifold is defined as a differentiable or smooth manifold \mathcal{S} together with a *Riemannian metric* k , forming the pair (\mathcal{S}, k) . The Riemannian metric k on \mathcal{S} defines at every point $\mathbf{q} \in \mathcal{S}$ an *inner product* $k_{\mathbf{q}}(\mathbf{x}, \mathbf{y})$ on the tangent space $T_{\mathbf{q}}\mathcal{S}$, with tangent vectors $\mathbf{x}, \mathbf{y} \in T_{\mathbf{q}}\mathcal{S}$. The inner product is positive definite for all tangent vectors, and determines lengths and angles. The length or norm induced by the inner product is given by $\|\mathbf{x}\|_2 := \sqrt{k(\mathbf{x}, \mathbf{x})} = \sqrt{\mathbf{x}^T \mathbf{K} \mathbf{x}}$, and the angle β between the two tangent vectors \mathbf{x} and \mathbf{y} can be defined by $\cos(\beta) \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 = k(\mathbf{x}, \mathbf{y})$.

The Riemannian metric is also referred to as the *metric tensor*. In local coordinates the metric tensor is expressed as the tensor of degree 2, or the symmetric positive definite matrix \mathbf{K} , also denoted $\mathbf{K}_{\mathbf{q}}$ to indicate that it is given at a point $\mathbf{q} \in \mathcal{S}$. A standard example of a Riemannian manifold is $(\mathcal{S}, k) = (\mathbb{R}^N, k_0)$, with k_0 the Euclidean inner product, and $\mathbf{K} = \mathbf{I}_N$. Besides angles and lengths of curves, the Riemannian metric allows for defining the area or curvature of a surface.

Curves and Geodesics

A *regular parametrized curve* is given as the continuously differentiable function $\gamma: [0, 1] \rightarrow \mathbb{R}^N$. $\dot{\gamma}(t_0)$ is the *tangent vector* at t_0 relative to γ , and $\dot{\gamma} \neq 0$ holds everywhere. The *length* of the curve is given as $\int_0^1 \|\dot{\gamma}(t)\|_2 dt$.

On a Riemannian manifold (\mathcal{S}, k) , the *length* of a curve $\gamma: [0, 1] \rightarrow \mathcal{S}$ can be written as $\int_0^1 \sqrt{k(\dot{\gamma}, \dot{\gamma})} dt$.

Geodesics are intuitively the generalization of straight lines to curved spaces. The shortest paths on a surface \mathcal{S} are known to be the geodesics. The geodesics can be found by locally minimizing the length of the curve γ on the surface \mathcal{S} . A geodesic is also the curve γ for which, at each point along the curve, the vector defined by the second derivative $\ddot{\gamma}$ is normal to the surface \mathcal{S} , i.e., a geodesic preserves the direction on a surface.

Surface Curvatures

For a unit normal vector at a point $\mathbf{q} \in \mathbb{R}^3$ of a surface $\mathcal{M} \subset \mathbb{R}^3$, a *normal plane* can be defined as a plane that contains the normal vector. The normal plane intersects the surface in a curve γ and contains the curve's tangent vector, $\dot{\gamma} \in T_{\mathbf{q}}\mathcal{M}$. Different normal planes at \mathbf{q} generate different such curves with different curvatures. The *principal curvatures* $k_{\mathbf{q},1}$ and $k_{\mathbf{q},2}$ of \mathcal{M} at \mathbf{q} result as the curvatures with maximum and minimum values, and the *principal curvature directions* are the directions of the corresponding tangent

vectors. The *mean curvature* can be defined based on the principal curvatures as $k_{\mathbf{q},\text{H}} = 1/2 (k_{\mathbf{q},1} + k_{\mathbf{q},2})$, which is an extrinsic measure of surface curvature. The *Gaussian curvature* is defined by $k_{\mathbf{q},\text{G}} = k_{\mathbf{q},1} k_{\mathbf{q},2}$ and is an example for an intrinsic measure of surface curvature. For $k_{\mathbf{q},\text{G}} = 0$, a *developable surface* results, which is a surface that can be flattened onto the plane without distortion. Examples for developable surfaces are cylinders and cones.

3.3 Computational Geometry

This section reviews two problems in computational geometry and describes algorithms that provide solutions. Our focus is mainly on Voronoi and Delaunay tessellations. In addition, we include some of the basics in the computation of shortest paths on graphs. The book by de Berg et al. (2000) serves us as an instrumental resource. Further information about Voronoi and Delaunay tessellations can be found in the survey paper by Du et al. (1999) and in the book on spatial tessellations by Okabe et al. (2000). More information on graph search methods and robot path planning are provided by Bertsekas (2005) and LaValle (2006).

3.3.1 Voronoi Tessellations

A Voronoi tessellation means a specific subdivision of space into so-called *Voronoi regions*, and is defined as follows.

Definition 3.1. A *Voronoi tessellation*¹ of an open set $\mathcal{W} \subset \mathbb{R}^N$ is the set of Voronoi regions $\mathcal{V}(\mathcal{P}) = \{V_i\}_{i=1}^n$, such that $n \geq 2$, $I(V_i) \cap I(V_j) = \emptyset$ for $i \neq j$, $\bigcup_{i=1}^n V_i = \mathcal{W}$, and

$$V_i = \{\mathbf{q} \in \mathcal{W} \mid \|\mathbf{p}_i - \mathbf{q}\|_2 \leq \|\mathbf{p}_j - \mathbf{q}\|_2, \forall j \in \{1, \dots, n\}, j \neq i\} \quad (3.4a)$$

$$= \bigcap_{i \neq j} H(\mathbf{p}_i, \mathbf{p}_j). \quad (3.4b)$$

The Voronoi regions represent polytopes in \mathcal{W} , each defined by half-spaces $H(\mathbf{p}_i, \mathbf{p}_j) = \{\mathbf{q} \in \mathcal{W} \mid \|\mathbf{p}_i - \mathbf{q}\|_2 \leq \|\mathbf{p}_j - \mathbf{q}\|_2\}$, for $i \neq j$.

The positions $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^n$ are the *generators* of the Voronoi tessellation. In the case where the generators are placed in a bounded domain $\Omega \subset \mathcal{W}$, a *bounded Voronoi tessellation* is defined as the set $\mathcal{V}(\mathcal{P}) = \{V_i \cap \Omega\}_{i=1}^n$. For

¹We will make use of the terms “Voronoi tessellation”, “Voronoi diagram” and “Voronoi partition” interchangeably along this thesis.

$n = 1$, there is only one generator and no Voronoi tessellation can be constructed; however, one might view the entire space \mathcal{W} , or Ω respectively, as the region associated with the single generator.

The dual of the Voronoi tessellation is the *Delaunay tessellation* $\mathcal{D}(\mathcal{P})$. Let us first look at the dual graph of $\mathcal{V}(\mathcal{P})$, denoted $G_{\mathcal{D}} = \{V_{\mathcal{D}}, E_{\mathcal{D}}\}$. The graph $G_{\mathcal{D}}$ has a node v_i for every Voronoi region V_i , and two nodes v_i and v_j are connected if the corresponding Voronoi regions V_i and V_j have a common edge or boundary, i.e., it holds $V_i \cap V_j \neq \emptyset$. This leads to a one-to-one correspondence between the bounded faces of $G_{\mathcal{D}}$ and the vertices of $\mathcal{V}(\mathcal{P})$. The Delaunay tessellation is then obtained from the dual graph $G_{\mathcal{D}}$ by embedding the edges $e_{ij} \in E_{\mathcal{D}}$ of $G_{\mathcal{D}}$ as the straight line segments $\overline{\mathbf{p}_i \mathbf{p}_j}$ into \mathbb{R}^N . In \mathbb{R}^2 , a *Delaunay triangulation* is obtained by adding additional edges to the 2D Delaunay tessellation. A triangulation is further called a Delaunay triangulation of $\mathcal{P} \subset \mathbb{R}^2$ if and only if the circumcircle of any triangle in the triangulation does not contain a point of \mathcal{P} in its interior (de Berg et al., 2000). Furthermore, an *optimal triangulation* can be defined in terms of the minimum angle over all its triangles, i.e., by how much the angles of the triangles in the triangulation differ. A Delaunay triangulation of \mathcal{P} is an angle-optimal triangulation, which maximizes the minimum angle over all triangulations of \mathcal{P} . Similar formulations can be obtained for N -dimensional Euclidean space by exchanging “triangles” and “circumcircles” for “polytopes” and “circum-hyperspheres”.

Definition 3.2. A *centroidal Voronoi tessellation (CVT)* is a Voronoi tessellation $\mathcal{V}(\mathcal{P})$, where the generators \mathcal{P} coincide with the centroids $\mathcal{C}_{\mathcal{V}} = \{\mathbf{c}_{V_i}\}_{i=1}^n$ of the Voronoi regions,

$$\mathbf{p}_i = \mathbf{c}_{V_i}. \quad (3.5)$$

The *centroid* or *mass center* \mathbf{c}_{V_i} of a Voronoi region V_i is given as

$$\mathbf{c}_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} \mathbf{q} \rho(\mathbf{q}) dF(\mathbf{q}), \quad M_{V_i} = \int_{V_i} \rho(\mathbf{q}) dF(\mathbf{q}), \quad (3.6)$$

where M_{V_i} is the *mass* and $\rho: \mathcal{W} \rightarrow \mathbb{R}_{\geq 0}$ is the (distribution) *density function*. An example for a CVT and its generation is provided by Figure 3.3 and Figure 3.4.

CVTs are related to optimization. As shown by Du et al. (1999), the CVT can be understood as the *local minimizer* of an *aggregate objective function* or *cost* of the form

$$\mathcal{H}(\mathbf{P}, \mathcal{Y}) = \sum_{i=1}^n \int_{Y_i} \|\mathbf{p}_i - \mathbf{q}\|_2^2 \rho(\mathbf{q}) dF(\mathbf{q}), \quad (3.7)$$

where the generators are written as a single nN -dimensional position vector, and $\mathcal{Y} = \{Y_i\}_{i=1}^n$ is an arbitrary tessellation of \mathcal{W} . In other words, the tessellation being the Voronoi tessellation, $\mathcal{Y} = \mathcal{V}$, and the generators being the centroids, $\mathbf{p}_i = \mathbf{c}_{V_i}$, is a necessary condition for $\mathcal{H}(\mathbf{P}, \mathcal{Y})$ to be minimized. In the case of a cost minimum, the generators at the centroids are the critical points of the objective function $\mathcal{H}_{\mathcal{V}}(\mathbf{P}) := \mathcal{H}(\mathbf{P}, \mathcal{V})$, $\mathbf{p}_i^* = \mathbf{c}_{V_i}$. The objective function $\mathcal{H}_{\mathcal{V}}(\mathbf{P})$ is nonconvex and a CVT represents a local minimum. CVTs are not unique. In \mathbb{R}^2 for example, regular tessellations into squares, triangles or hexagons all represent valid instances of CVTs.

However, a global minimizer does exist, and the *optimal* CVT leads to the lowest value of $\mathcal{H}_{\mathcal{V}}(\mathbf{P})$ among all the possible CVTs. *Gershon's conjecture* states that “asymptotically as the number of generators gets larger and larger, the optimal CVT will be forming a regular tessellation consisting of the replication of a single polytope whose shape depends only on the spatial dimension” (Du and Wang, 2005a). For two dimensions, such a single polytope is known and the basic Voronoi region of an optimal CVT is the regular hexagon.

Computation of Voronoi Tessellations

An optimal algorithm for computing Voronoi tessellations is known as *Fortune's algorithm*, which runs for n generators in $O(n \log n)$ time (de Berg et al., 2000). The computation of CVTs can be achieved by probabilistic or deterministic algorithms: probabilistic *MacQueen's algorithm* is based on random sequential sampling, whereas deterministic *Lloyd's algorithm* computes Voronoi tessellations and centroids iteratively. Our focus is on the Lloyd's algorithm (Lloyd, 1982).

A description of Lloyd's algorithm is given in Algorithm 1. Lloyd's algorithm basically consists of two steps: (1) the computation of a Voronoi tessellation for the current generator set, which assigns regions to the generators, and (2) the update of the generators with the regions' centroids, which minimizes the objective function $\mathcal{H}_{\mathcal{V}}(\mathbf{P})$ further. Lloyd's algorithm can be analyzed under different viewpoints. As we will also see in Section 3.5.3, in its simplest form it implements a *gradient descent method*, which results in linear convergence to the final CVT. Moreover, through its iterative nature and convergence of the generators to a fixed final configuration of centroids $\mathcal{C}_{\mathcal{V}} = \{\mathbf{c}_{V_i}\}_{i=1}^n$, it represents a *fixed point iteration*. Let in N dimensions be $L: \mathbb{R}^{nN} \rightarrow \mathbb{R}^{nN}$, with $L = [L_i]_{i=1}^n$ and $L_i(\mathbf{P}) = \mathbf{c}_{V_i}$ for the nN -dimensional position vector $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n$. It follows immediately that CVTs are fixed points of $L(\mathbf{P})$.

Algorithm 1 Lloyd's Algorithm

Require: Set of n generators with initial positions $\mathcal{P} = \{\mathbf{p}_i^0\}_{i=1}^n \subset \Omega$, for the given set $\Omega \subset \mathcal{W}$, with density function ρ defined on \mathcal{W} .

- 1: **loop** {Fixed point iteration}
 - 2: Construct the Voronoi tessellation \mathcal{V} of Ω with respect to \mathcal{P}
 - 3: Compute the centroids $\mathcal{C}_{\mathcal{V}}$ of the Voronoi regions \mathcal{V} ;
 update the locations with these centroids: $\mathcal{P} \leftarrow \mathcal{C}_{\mathcal{V}}$
 - 4: **end loop**
 - 5: **return** Final configuration $\{\mathcal{V}, \mathcal{C}_{\mathcal{V}}\}$ upon convergence
-

Generalizations of Voronoi Tessellations

Over the years, many variants of Voronoi tessellations have been developed, which lead to generalizations of the original Voronoi and Delaunay tessellations. A Voronoi tessellation can basically vary in the set of generators used, the way we measure distance between generators, and the type of space we consider. In our case, we look exclusively at point sets \mathcal{P} , use a distance function $d(\mathbf{q}, \mathbf{p}_i)$ which is either based on the Euclidean distance or a shortest path distance, and study Euclidean space as well as curved spaces. Note that a consistent generalization of a Voronoi tessellation or CVT should always reduce to the original Voronoi tessellation or CVT under appropriate settings. We give here three examples of extensions to CVTs, which are of relevance for the subsequent chapters. Further examples are provided by Du et al. (1999) and Okabe et al. (2000).

Constrained CVTs. Related to the definition of a CVT, we can define the *constrained centroidal Voronoi tessellation (CCVT)* as the Voronoi tessellation $\mathcal{V}(\mathcal{P})$ with Voronoi regions

$$V_{C,i} = \{\mathbf{q} \in \mathcal{S} \mid \|\mathbf{p}_i - \mathbf{q}\|_2 \leq \|\mathbf{p}_j - \mathbf{q}\|_2, \forall j \in \{1, \dots, n\}, j \neq i\}, \quad (3.8)$$

restricted to $\mathcal{S} = \{\mathbf{q} \in \mathbb{R}^N \mid G_0(\mathbf{q}) = 0 \wedge G_j(\mathbf{q}) \leq 0, \forall j \in \{1, \dots, M\}\}$, with continuous functions $G: \mathbb{R}^N \rightarrow \mathbb{R}$, and a set of generators $\mathcal{P} \subset \mathcal{S}$. The positions $\mathbf{p}_i \in \mathcal{P}$ are located at the *constrained centroids* $\mathbf{c}_{V_{C,i}}$ of the Voronoi regions $V_{C,i}$. They are found to be the normal projections \mathbf{c}'_{V_i} of the unconstrained centroids \mathbf{c}_{V_i} , defined by Equation (3.6) in \mathbb{R}^N , onto \mathcal{S} (Du et al., 2002). Du et al. (2002) shows that the CCVT is a necessary condition for the minimization of the aggregate objective function after Equation (3.7) under constraints, where generators are positioned at

$\mathbf{p}_i^* = \mathbf{c}_{V_{C,i}} = \mathbf{c}'_{V_i}$. Hence, the constrained centroids are further given by $\mathbf{c}_{V_{C,i}} = \operatorname{argmin}_{\mathbf{p}_i \in V_{C,i}} \int_{V_{C,i}} \|\mathbf{p}_i - \mathbf{q}\|_2^2 \rho(\mathbf{q}) dF(\mathbf{q})$.

Discrete CVTs. Given a discrete set of points $\mathcal{Q} \subset \mathbb{R}^N$, the *discrete centroidal Voronoi tessellation (DCVT)* in Euclidean space can be defined as the Voronoi tessellation $\mathcal{V}(\mathcal{P})$ with Voronoi regions

$$V_{D,i} = \{\mathbf{q} \in \mathcal{Q} \mid \|\mathbf{p}_i - \mathbf{q}\|_2 \leq \|\mathbf{p}_j - \mathbf{q}\|_2, \forall j \in \{1, \dots, n\}, j \neq i\}, \quad (3.9)$$

where the generators are located at the centroids $\mathbf{c}_{V_{D,i}}$. Points that are equidistant to two or more generators are assigned according to a predefined priority; we assign point \mathbf{q} to the generator with lowest index i , i.e., $\min\{i, j\}$ for $\|\mathbf{p}_i - \mathbf{q}\|_2 = \|\mathbf{p}_j - \mathbf{q}\|_2$. The positions $\mathbf{p}_i \in \mathcal{P}$ can either be contained in the full set \mathbb{R}^N or are restricted to the discrete set \mathcal{Q} , which leads to *discrete centroids* $\mathbf{c}_{V_{D,i}} \in \mathcal{Q}$. The aggregate objective function now becomes $\mathcal{H}_{\mathcal{V}}(\mathbf{P}) = \sum_{i=1}^n \sum_{\mathbf{q} \in V_{D,i}} \|\mathbf{p}_i - \mathbf{q}\|_2^2 \rho(\mathbf{q})$ and the centroids can be computed as $\mathbf{c}_{V_{D,i}} = \operatorname{argmin}_{\mathbf{P}_i} \sum_{\mathbf{q} \in V_{D,i}} \|\mathbf{p}_i - \mathbf{q}\|_2^2 \rho(\mathbf{q})$. Note that this formulation of the Voronoi tessellation and objective function is typically used in clustering.

Another discrete version of a CVT can be defined over a graph. The Voronoi tessellation $\mathcal{G}(V_G) = \{V_{G_i}\}_{i=1}^n$ of a graph $G = \{V_G, E_G\}$ is also called the *graph Voronoi tessellation* or *network node Voronoi diagram* (Okabe et al., 2000). A DCVT on the graph is then given by the graph Voronoi tessellation $\mathcal{G}(V_G)$, where each Voronoi region V_{G_i} corresponds to the set of vertices of the subgraph $G_i = \{V_{G_i}, E_{G_i}\}$, $G_i \subset G$, with

$$V_{G_i} = \{v \in V_G \mid d_G(v, v_{\mathbf{p}_i}) \leq d_G(v, v_{\mathbf{p}_j}), \forall j \in \{1, \dots, n\}, j \neq i\}, \quad (3.10)$$

and each generator related to vertex $v_{\mathbf{p}_i} \in V_{G_i}$ coincides with the *graph centroid* indicated by vertex $v_{\mathbf{c}_i} \in V_{G_i}$. The edges of the subgraph G_i are given by the set $E_{G_i} = \{e = (v, w) \mid e \in E_G \wedge v, w \in V_{G_i}, v \neq w\}$. Similar to the example above, we prioritize vertices of lowest index if several vertices are equidistant, i.e., $\min\{i, j\}$ for $d_G(v, v_{\mathbf{p}_i}) = d_G(v, v_{\mathbf{p}_j})$. This time the aggregate objective function becomes $\mathcal{H}_{\mathcal{G}}(V_G) = \sum_{i=1}^n \sum_{v \in V_{G_i}} d_G(v, v_{\mathbf{p}_i})^2 \rho_G(v)$, which is a function of the vertex set of discrete generator positions $\{v_{\mathbf{p}_i}\}_{i=1}^n$, and we get for the graph centroids $v_{\mathbf{c}_i} = \operatorname{argmin}_{v_{\mathbf{p}_i} \in V_{G_i}} \sum_{v \in V_{G_i}} d_G(v, v_{\mathbf{p}_i})^2 \rho_G(v)$.

The distance $d_G: V_G \times V_G \rightarrow \mathbb{R}_{\geq 0}$ is the (weighted) shortest path distance on the graph G (possibly weighted by edge weights $\omega(e)$), and $\rho_G: V_G \rightarrow \mathbb{R}_{\geq 0}$ denotes the weight at vertex v .

CVTs in Non-Euclidean Metrics. In this last example, we present a more abstract variant of a Voronoi tessellation using a non-Euclidean metric. The Voronoi tessellation of a curved space obeys—different from tessellations of Euclidean space—the laws of Riemannian geometry (see Section 3.2). For example, a curved surface in \mathbb{R}^3 , such as a sphere or torus, can be described by a parametrization in the 2D parameter plane; computing a CVT on the plane in non-Euclidean metric and mapping back onto the curved surface may result in a conventional CVT of the parametric surface.

The *anisotropic centroidal Voronoi tessellation (ACVT)* is defined according to Du and Wang (2005b) as the Voronoi tessellation $\mathcal{V}(\mathcal{P})$ of the 2D Riemannian manifold \mathcal{S} , with the Riemannian metric described as the positive definite metric tensor $\mathbf{K}_{\mathbf{q}}$ and the Voronoi region given as

$$V_{A,i} = \{ \mathbf{q} \in \mathcal{S} \mid d_{\mathbf{q}, \mathbf{p}_i} \leq d_{\mathbf{q}, \mathbf{p}_j}, \forall j \in \{1, \dots, n\}, j \neq i \}, \quad (3.11)$$

such that $\mathbf{p}_i = \mathbf{c}_{V_{A,i}}$, i.e., the generators $\mathbf{p}_i \in \mathcal{P}$ are positioned at the *anisotropic centroids* $\mathbf{c}_{V_{A,i}}$. Du and Wang (2005b) suggests to measure distance by a *directional distance* $d_{\mathbf{q}, \mathbf{p}_i} = \sqrt{(\mathbf{p}_i - \mathbf{q})^T \mathbf{K}_{\mathbf{q}} (\mathbf{p}_i - \mathbf{q})}$. Although this distance is not symmetric and thus no proper metric as such, it proves adequate for a practical definition of the ACVT. It is consistent with the definition of standard Voronoi diagrams in the isotropic metric and straightforward to compute (see Du and Wang (2005b) for further discussions). A necessary condition for the optimization problem of minimizing cost $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$ over the curved space \mathcal{S} requires the tessellation of \mathcal{S} to be an ACVT (Du and Wang, 2005b). The anisotropic centroids can again be computed from the minimization of the aggregate objective function $\mathcal{H}_{\mathcal{V}}(\mathcal{P}) = \sum_{i=1}^n \int_{V_{A,i}} d_{\mathbf{q}, \mathbf{p}_i}^2 \rho(\mathbf{q}) dF(\mathbf{q})$.

Applying the directional distance, we get $\mathbf{c}_{V_{A,i}} = \left(\int_{V_{A,i}} \mathbf{K}_{\mathbf{q}} \rho(\mathbf{q}) dF(\mathbf{q}) \right)^{-1} \int_{V_{A,i}} \mathbf{K}_{\mathbf{q}} \mathbf{q} \rho(\mathbf{q}) dF(\mathbf{q})$, for each $V_{A,i}$. Finally note that, for an isotropic metric tensor $\mathbf{K}_{\mathbf{q}}$, the ACVT reduces to the original CVT with Euclidean distance metric.

3.3.2 Graph Search and Shortest Paths

A common way to represent the geometry of an environment is by a graph $G = \{V, E\}$. Classical graph-based representations from computational geometry include Delaunay graphs, Voronoi diagrams, as well as *reduced visibility graphs* (LaValle, 2006). These graphs all define so-called *roadmaps*. Roadmaps are topological graphs, which describe the connectivity information of an environment's geometry. Roadmaps are, besides regular 2D grids, typical environment representations in robot path planning.

A (discrete) shortest path from a start vertex v_s to a goal vertex v_g on the graph G is computed by a *graph search* or *label correcting method* (Bertsekas, 2005). Graph search methods assign a cost to each encountered vertex v ,

$$f(v) = g(v) + h(v). \quad (3.12)$$

The cost function g denotes the *cost-to-come*, i.e., the accumulated cost along the path from the start vertex v_s to the current vertex v . The cost function h represents the *cost-to-go*, i.e., the estimated cost from the current vertex v to the goal vertex v_g . As the optimal cost-to-go is not known, a *heuristic* is used. Thereby, the heuristic must always be *admissible* in order to guarantee optimal shortest paths, which means that the heuristic is always an underestimate of the true optimal cost-to-go (LaValle, 2006). A common choice for the heuristic h is the Euclidean distance between v and v_g .

The known vertices are stored in a *priority queue* L . The vertices are sorted according to their cost value $f(v)$ and at each iteration of the search, the vertex with lowest cost is selected next. The selected vertex is removed from the priority queue and expanded, i.e., its neighboring vertices are inserted into the priority queue. If a neighboring vertex has already been inserted into the priority queue, it is not added again but its cost value $f(v)$ is updated. The graph search method terminates when the goal vertex is expanded.

The following three graph search methods are popular in robotics, and particularly relevant for this thesis:

Dijkstra's Algorithm. The *Dijkstra's algorithm* (Dijkstra, 1959) is a systematic graph search method that does not use a heuristic estimate. The cost reduces to $f(v) = g(v)$. Dijkstra's algorithm guarantees an optimal solution.

A* Algorithm. The *A* algorithm* (Hart et al., 1968) is the heuristic version of the Dijkstra's algorithm and expands, just as described above, the vertices in the priority queue according to the smallest value $f(v) = g(v) + h(v)$. The A* algorithm also results in an optimal solution.

D* Algorithm. The *D* algorithm* (Koenig and Likhachev, 2002; Stentz, 1995) is an incremental version of the A* algorithm that allows for efficient replanning. If a graph undergoes a dynamic update, i.e., vertices or edges, or vertex or edge cost respectively, have changed, the D* algorithm does not need to plan completely anew. Only edges or vertices that were affected by the change in the graph are reevaluated. This results in an overall improved search time during replanning.

3.4 Structure Inference

This section gives a brief introduction to *tensor voting*, a method for perceptual grouping and structure inference from sparse and noisy data. A more general overview of the tensor voting framework can be found in the books by Medioni et al. (2000) and Mordohai and Medioni (2006). In addition, King (2008) provides valuable insight into the application of tensor voting to 3D environment modeling. Tensor voting can be used for the preprocessing of LIDAR point clouds in the navigation solutions of Chapter 7 and Chapter 8.

3.4.1 Basics of Tensor Voting

Tensor voting is a generic solution for perceptual organization problems, and is based on a set of *Gestalt principles*, which are believed to be used by the human visual system (Mordohai and Medioni, 2006). Tensor voting is particularly useful because it can be applied to a wide range of applications and data types. Moreover, it can be performed in any number of dimensions to segment *input tokens*, such as points in a point cloud, into distinct structures. For example, in two dimensions, input can be grouped into curve-like structures (which have an associated direction) and regions or junctions (which have no associated direction). Similarly, in three dimensions, input can be grouped into surfaces (which have two associated directions), curves (with one preferred direction), and directionless points or regions (characteristic of noise or volumes). Curve elements can be found at the intersection of two surfaces, which, for example, proves extremely useful for classifying step-like obstacles in the environment of a climbing robot like MagneBike.

As mentioned, in three dimensions, there are the three fundamental types of structures: *surfaces*, *curves*, and unoriented *points*. Information needed for voting and segmenting into these fundamental structures is encoded as 3D symmetric tensors of degree 2. This can be conceptualized as a 3×3 matrix, or alternatively as an ellipsoid where the eigenvectors of the matrix represent the axes of the ellipsoid, and the eigenvalues represent the size of the ellipsoid in each corresponding direction. There are therefore also three fundamental types of tensors—“stick”, “plate”, and “ball” tensors—signifying the three aforementioned structures, as shown in Figure 3.1. These tensors are aligned with the normal space of the corresponding surface. Therefore, surface elements are represented by a 1D stick tensor, aligned with the surface normal. Curves or edges, on the other hand, are given by 2D plate tensors, where the tangent direction is given by the third eigenvector, which has a corresponding null eigenvalue. Tokens with no associated direction are

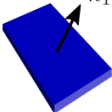

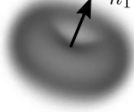
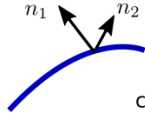
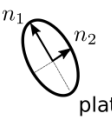
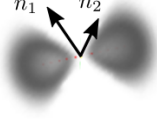
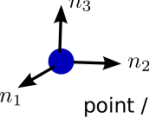

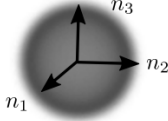
Structure	Tensor	Tensor Field
 surface	 stick $\lambda_1 = \lambda$ $\lambda_2 = \lambda_3 = 0$	
 curve	 plate $\lambda_1 = \lambda_2 = \lambda$ $\lambda_3 = 0$	
 point / region	 ball $\lambda_1 = \lambda_2 = \lambda_3 = \lambda$	

Figure 3.1: Tensor voting in three dimensions. The fundamental types of structures are shown in the left-most column. The second column shows the form of the representative 3D tensors, with their eigenvalues λ_i , $i = \{1, 2, 3\}$. The last column shows the form of the voting fields cast by each fundamental structure. The vectors representing the structures' normal spaces are illustrated in each column as well. (Adapted from Mordohai and Medioni (2006).)

encoded as unoriented 3D symmetric ball tensors. In general, the magnitudes of the eigenvalues designate the associated *salience*, and any arbitrary tensor can be decomposed into its ball, plate and stick components in order to evaluate the dominance of each type of structure at that location. In three dimensions, the decomposition is achieved in accordance with Mordohai and Medioni (2006) by

$$T = \lambda_1 \hat{\mathbf{e}}_{x,W} \hat{\mathbf{e}}_{x,W}^T + \lambda_2 \hat{\mathbf{e}}_{y,W} \hat{\mathbf{e}}_{y,W}^T + \lambda_3 \hat{\mathbf{e}}_{z,W} \hat{\mathbf{e}}_{z,W}^T \quad (3.13a)$$

$$\begin{aligned}
 &= (\lambda_1 - \lambda_2) \hat{\mathbf{e}}_{x,W} \hat{\mathbf{e}}_{x,W}^T + && \text{(stick component)} \\
 &(\lambda_2 - \lambda_3) (\hat{\mathbf{e}}_{x,W} \hat{\mathbf{e}}_{x,W}^T + \hat{\mathbf{e}}_{y,W} \hat{\mathbf{e}}_{y,W}^T) + && \text{(plate component)} \\
 &\lambda_3 (\hat{\mathbf{e}}_{x,W} \hat{\mathbf{e}}_{x,W}^T + \hat{\mathbf{e}}_{y,W} \hat{\mathbf{e}}_{y,W}^T + \hat{\mathbf{e}}_{z,W} \hat{\mathbf{e}}_{z,W}^T) && \text{(ball component)}
 \end{aligned} \quad (3.13b)$$

with λ_i , $i \in \{1, 2, 3\}$, representing the eigenvalues in order of decreasing magnitude, and $\hat{\mathbf{e}}_{x,W}$, $\hat{\mathbf{e}}_{y,W}$, $\hat{\mathbf{e}}_{z,W} \in \mathbb{R}^3$ representing the corresponding eigenvectors with respect to the world frame W .

Through voting, each component of the tensor is able to propagate its information according to specific voting fields. The idea is for votes to convey

their apparent structure as if they were smoothly continued to the location where a vote takes place. The strength of the vote is based on the likely prevalence of the hypothetical structure at that location. Votes are cast by every input token, e.g., points in a point cloud, and collected using simple tensor addition. Ball tensors have no preferred orientation, and therefore propagate uniformly in all directions, with a ball shaped tensor field. Plate tensors, which represent curve elements, have a roughly stick shaped tensor field, as the structure is likely to continue only along the tangent directions. Stick tensors, on the other hand, vote with a plate shaped tensor field in order to continue the surface-like structure. An illustration of the various tensor field forms is included in Figure 3.1. The scale of these voting fields, denoted by the *voting scale parameter* σ , can be chosen based on the approximate feature size.

The general framework of the tensor voting algorithm, as proposed by Medioni et al. (2000), is shown in Figure 3.2 on the left. To start, the sparse input tokens are transformed into tensors, which encode any known saliency or directionality. If no preliminary structural information is known, the input tokens are simply encoded as unit ball tensors. Once the sparse tensors are initialized, they are refined through a round of sparse voting. During *sparse voting*, all input tokens cast votes only at the locations of all other input tokens. The decomposed components of the sparse tensors are then used to densify the information through another round of voting, so-called *dense voting*, where votes are cast by all input tokens throughout the entire space, e.g., represented by a 3D grid². Once the final votes have been collected and added up, the components of the tensors can be used to analyze the underlying structures.

3.4.2 Tensor Voting Applied to Point Clouds

When originally described by Medioni et al. (2000), tensor voting was intended for computer vision applications, where everything is usually in two dimensions, dense and uniformly distributed. The framework could therefore benefit from a few modifications when applied to different types of applications, such as inference from LIDAR point clouds. A few changes in the tensor voting framework are discussed, which were suggested by Mordohai and Medioni (2010) and King (2008). The first change is related to an analytic formulation of the voting fields, the second change proposes a different vote decay function, and the third change addresses methods of token reduction.

²Note that, as depicted in Figure 3.2 on the left, ball components will not vote in this round in our case, as our analysis is focused on surfaces and edges rather than regions.

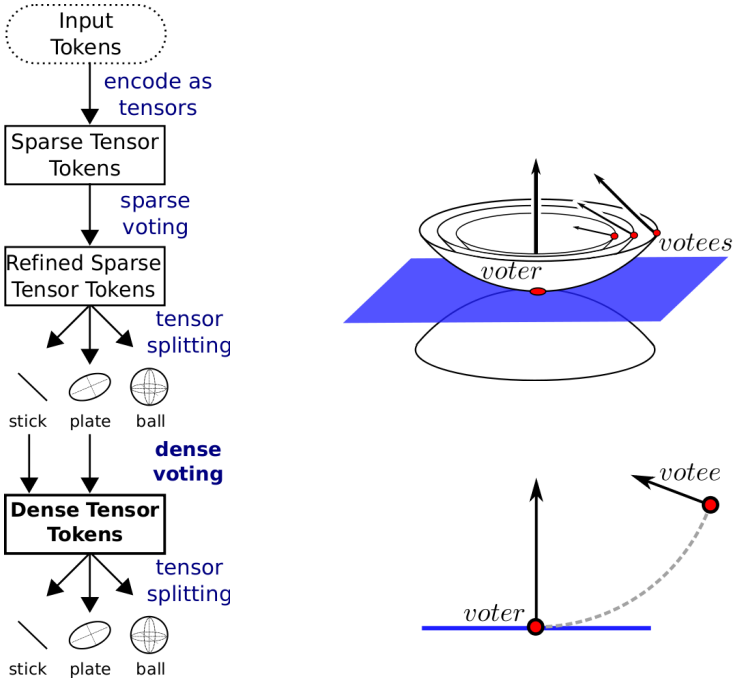


Figure 3.2: Tensor voting framework. Left: Sequence of sparse and dense voting, as well as tensor splitting, decomposing the tensor into its structural components. (Adapted from Medioni et al. (2000).) Right: Formulation of the stick vote in three dimensions. The osculating spheres represent where the plate shaped tensor field spreads from the voter. The vectors at the votees (locations where the votes are cast) show the directions of the votes, which are normal to the osculating spheres.

Calculation of Voting Fields

A vote cast by a tensor is meant to impart its contained structure, as if the structure was continued smoothly toward the location of the vote. For a stick vote, the assumption is made that the smoothest continuation of a surface is described by an arc of constant curvature which runs tangent to the surface at the voter's location, or in other words, an osculating sphere. Therefore, a vote cast by a stick voter is also a stick, with the direction that is normal to the osculating sphere at that location (see Figure 3.2, right). In the original implementations of tensor voting, the plate and ball fields

were created by integrating the stick field as it is rotated about one or two of its axes, respectively. These integrals unfortunately have no closed form solution, and so the integration is typically done numerically by rotating the stick tensor by fixed steps and adding the results from each position. The resulting field is then normalized, such that its energy is equivalent to that from the stick field. The results are stored in a look-up table at the desired resolution, and then during the voting procedure, values are obtained by interpolating between fields in the look-up table.

Calculation of votes using these precomputed look-up tables can be computationally costly and inaccurate, especially when the number of dimensions is increased. New analytic formulations for the voting fields can be found in the works of Mordohai and Medioni (2010) and King (2008), which lead to improvements in the generation and storage of voting fields for three and higher dimensions in particular.

Vote Decay Function

In general, the magnitude of a vote is modulated based on how likely the continuation of the hypothetical structure is. This can be evaluated based on two of the Gestalt principles, which are *proximity* and *smooth continuation* (Mordohai and Medioni, 2006). Therefore, the vote weighting can be represented as a function of distance for proximity and curvature for smooth continuation. Typically, saliency decays exponentially with distance squared. As a result, two tokens placed at almost the exact same location will produce extremely strong votes for each other, despite supplying very little new information regarding the local structure. This can easily lead to amplification of small-scale noise, as well as introduce large effects from density variations. In addition, the commonly used curvature decay function was designed for computer vision applications, and therefore exhibits strange behavior when distances are not defined in pixels. For example, the shape of the decay function changes drastically as the voting scale parameter σ is varied, and is completely invalid for σ values less than one.

A new weighting function for votes was derived by King (2008), which solves the aforementioned issues. The intuition used in the derivation of the new decay function is that tokens should have the strongest votes at a distance of σ , as this is the expected feature scale. Any votes much closer or much farther than σ will receive a lower weighting. Additionally, the curvature decay function and the distance decay function are decoupled, eliminating unwanted side effects caused by the choice of σ . This new way of choosing the decay function has also been shown to provide smoother and more accurate results (King, 2008).

Token Reduction

Typically, LIDAR point clouds contain a large number of points, with sizable density variations. These density variations can skew the saliency results from tensor voting in favor of densely sampled regions. Although there should be some preference for detecting structures at denser regions, the emphasis on these areas is usually too strong, preventing any structure inference at the less dense regions. The input tokens are therefore subsampled in order to unify the density and reduce the computation time. This is achieved in the work of King (2008) by a multi-scale tensor voting method. We present an adapted and simplified method for token reduction in Section 8.2.2.

3.5 Control Theory

Finally, this section summarizes some relevant concepts from control theory, which are helpful for the understanding of the following chapters. We mainly adhere to the terminology found in the books on robot motion planning and control by Laumond (1998), on planning algorithms by LaValle (2006), on optimal control by Bertsekas (2005), on cooperative control by Shamma (2008) and on distributed control of robotic networks by Bullo et al. (2009), where the interested reader will find further detail.

3.5.1 System Models

The general system model we use to describe a robot is given as the *state transition equation*

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad (3.14)$$

with state $\mathbf{x} \in \mathcal{X}$ and control input $\mathbf{u} \in \mathcal{U}$.

The motion of a robot is then expressed as the sequence of states, which results in a path or trajectory.

Definition 3.3. A *trajectory* $\sigma: \mathcal{X} \times \mathcal{U} \times \mathbb{R} \rightarrow \mathcal{X}$ relates the control input $\mathbf{u}(t)$ through a system model $f(\mathbf{x}(t), \mathbf{u}(t))$ to the state $\mathbf{x}(t)$ evolving over time t , where $\mathbf{x}_0 = \mathbf{x}(0)$ is the initial state of the system,

$$\sigma(\mathbf{x}_0, \mathbf{u}, t) = \int_0^t f(\mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau + \mathbf{x}_0. \quad (3.15)$$

Note that a trajectory with the state space \mathcal{X} limited to the position subspace, e.g., given by the (x, y, z) -coordinates in the 3D workspace, defines a *path*. Moreover, the trajectory limited to the 6D state space, resulting from

the full pose in the 3D workspace, which includes the (x, y, z) -coordinates of the position and pitch, roll and yaw angles of the orientation, will be called a *6-DoF path* throughout this thesis.

The goal of path or motion planning is to generate a path or trajectory for a robot, such that it is collision-free or feasible and preferably optimal, i.e., feasible while also complying with some quality criteria.

Definition 3.4. A *feasible path* is a path $\gamma: [0, 1] \rightarrow \Omega$, for the free space Ω , such that $\gamma(0) = \mathbf{p}^0$ is the start position and $\gamma(1) = \mathbf{g}$ is the goal position of the robot.

Definition 3.5. An *optimal path* is a path $\gamma^* = \underset{\gamma \in \Gamma}{\operatorname{argmin}} \mathcal{F}(\gamma)$ for a given cost function $\mathcal{F}: \Gamma \rightarrow \mathbb{R}_{\geq 0}$ and the set of all paths Γ , such that γ and thus γ^* are feasible.

Global path planning focuses on the planning of a global path from a start to a goal region through the environment, i.e., a path which is of low resolution and may even consist of a sequence of discrete path segments. In contrast, *local path planning* generates a local path, i.e., a path along a subpart or segment of the global path, and focuses on collision avoidance and the actual motion planning of the robot.

A path planning algorithm is *complete* if it is able to determine, for any input, if a feasible path for the robot exists (LaValle, 2006).

In robot systems, continuous and discrete or logical processes are often to be combined. *Hybrid systems* offer a system model to describe the interactions between continuous and discrete dynamics. In the broader sense, we will refer to hybrid systems when describing coverage algorithms in Chapter 4; our coverage algorithms combine different dynamics and switch between different coverage states.

Unicycles and Differential-Drives, Bicycles and Car-Like Robots

For the methods presented in the following chapters it is sufficient to capture the robots' kinematics and model the systems by kinematic models.

A holonomic point robot can be modeled by a *single integrator*,

$$\dot{\mathbf{x}}(t) = \mathbf{u}(t), \quad (3.16)$$

where $\mathbf{x} = \mathbf{p}$. We use the single integrator model in our coverage solutions at the high level of the system architecture.

The e-puck as well as the MagneBike robots (see Chapter 2.2) are *nonholonomic*, i.e., the admissible motions are subject to nonholonomic constraints, caused by the “rolling without slipping” condition between the wheels and the ground (Laumond, 1998). In addition, both robot platforms are able to move backward and to turn on the spot, which can come in useful if a robot has to maneuver in narrow spaces or faces a dead end. Turning on the spot furthermore allows for the implementation of bang-bang control-like behaviors.

The e-puck robot can be modeled as a *unicycle* or *differential-drive* vehicle in 2D workspace through the state transition equation

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}(t), \quad (3.17)$$

with state $\mathbf{x}(t) = [x(t) \ y(t) \ \theta(t)]^T \in \mathbb{R}^2 \times \mathbb{S}^1$, and control input $\mathbf{u}(t) = [v(t) \ \omega(t)]^T \in \mathbb{R}^2$ set to the linear and angular velocity input. The (x, y) -coordinates denote the position of a reference point of the robot in 2D workspace with respect to a Cartesian world frame W , and the heading angle θ is defined as the angle enclosed by the positive x -axes of the world frame W and the local robot body frame B .

The MagneBike robot can be approximated by a simplified bicycle or car-like model, i.e., there is a single front and rear wheel attached to the midpoint of the front and rear axle of the robot and only the front wheel can be steered. In case of MagneBike, both the front and the rear wheel are actively driven but the wheel units are coupled by a deformation controller, which matches the velocities of the front and the rear wheel³. As a consequence of this, the MagneBike robot can be modeled as *front-wheel-drive bicycle* or *rear-wheel-drive bicycle*, depending on what is beneficial for a given application or controller design.

In case of rear-wheel driving, the system model is defined by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ \tan(\phi(t))/L & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}(t), \quad (3.18)$$

with state $\mathbf{x}(t) = [x(t) \ y(t) \ \theta(t) \ \phi(t)]^T \in \mathbb{R}^2 \times (\mathbb{S}^1)^2$ describing the position of the rear wheel, heading of the robot body as well as steering angle

³For further details of the robot’s low-level controllers, refer to Tâche (2010).

of the front wheel, and control input $\mathbf{u}(t) = [v_{\text{rear}}(t) \quad \varsigma(t)]^T \in \mathbb{R}^2$, given as the rear wheel's driving velocity and the front wheel's steering velocity input. L denotes the distance between the front and the rear wheel.

If the robot is modeled with front-wheel driving, we get instead

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \cos(\phi(t)) \cos(\theta(t)) & 0 \\ \cos(\phi(t)) \sin(\theta(t)) & 0 \\ \sin(\phi(t))/L & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}(t), \quad (3.19)$$

where the control input $\mathbf{u}(t) = [v_{\text{front}}(t) \quad \varsigma(t)]^T \in \mathbb{R}^2$ now is given as the front wheel's driving and steering velocity input. The front-wheel-drive model from Equation (3.19) is equivalent to the kinematic model of a differential-drive robot pulling a trailer, which represents the rear wheel unit of the car. Alternatively, the same equation can be expressed in the simpler form of Equation (3.17) as a differential-drive robot if the direction of the front wheels are not explicitly considered and the control input u_2 is directly assumed as $\phi(t)$ instead of $\varsigma(t) = \dot{\phi}(t)$. For Equation (3.17), the velocity of the reference point on the rear wheel, $v_{\text{rear}}(t)$, and the angular velocity around the rear wheel, $\omega(t)$, are obtained by an input transformation as the new control inputs of the system. The original control input $\mathbf{u}(t) = [v_{\text{front}}(t) \quad \phi(t)]^T \in \mathbb{R}^2$ is related to $v_{\text{rear}}(t)$ and $\omega(t)$ through

$$\phi(t) = \cot(\omega(t) L / v_{\text{rear}}(t)), \quad (3.20a)$$

$$v_{\text{front}}(t) = v_{\text{rear}}(t) / \cos(\phi(t)). \quad (3.20b)$$

The main difference of the reformulation of a front-wheel-drive as a differential-drive model in comparison to the real differential-drive robot is in the admissible controls, i.e., the feasible paths of the modeled bicycle robot are bound in curvature by a maximum value of $1/L$ (Laumond, 1998).

3.5.2 Gradient Descent Controller

A *control law* or *controller* of a robot applies control inputs from the control space \mathcal{U} to the system model of Equation (3.14). For real applications, open-loop control is often not satisfactory due to unmodeled disturbances in the robot's motion or environment, and due to noise in the robot's sensing. A *state-feedback controller* observes the state of the system and can track a trajectory during execution. Feedback control defines the control *policy* $\pi: \mathcal{X} \rightarrow \mathcal{U}$.

The *gradient descent method* is defined by the iterative algorithm $\mathbf{x}_{k+1} = \mathbf{x}_k - c_k \nabla f(\mathbf{x}_k)^\top$, with $f: \mathbb{R}^N \rightarrow \mathbb{R}$ and positive step size c_k (Luenberger and Ye, 2008).

A *gradient descent controller* can be defined for positive gain k_1 as

$$\mathbf{u}(t) = \pi(\mathbf{x}(t)) = -k_1 \nabla \mathcal{F}(\mathbf{x}(t))^\top, \quad (3.21)$$

with the cost function $\mathcal{F}: \mathcal{X} \rightarrow \mathbb{R}$ defined over the state space. This is directly related to potential functions and navigation functions (LaValle, 2006). We can also write more generally for the positive control gains k_1 and k_2

$$\mathbf{u}(t) = k_1 \mathbf{z}(t) = k_2 \widehat{\mathbf{z}}(t), \quad (3.22)$$

where \mathbf{z} is given as a function of the negative gradient, $-\nabla \mathcal{F}$.

3.5.3 Cooperative Control

Shamma (2008) describes cooperative control concisely as the control of an overall system that is comprised of “*a collection of decision-making components with limited processing capabilities, locally sensed information, and limited inter-component communication, all seeking to achieve a collective objective*”. Throughout this thesis we will come across several systems and algorithms which fall into the category of cooperative control, all exhibiting a subset of these characteristic properties.

Voronoi Coverage

A common objective in the coordination of multiple robots is the cooperative deployment and coverage of an environment. Cortés et al. (2004) introduced a motion coordination method for coverage control with mobile sensor networks, which builds on aggregate objective functions from locational optimization. As a direct consequence, the method exhibits strong links to centroidal Voronoi tessellations, and we refer to Section 3.3.1 repeatedly throughout this section. We revisit the most important concepts of the coverage method by Cortés et al. (2004) in the following, since it lays the foundation for a large part of the developments in this thesis. We will refer to the method as *Voronoi coverage* from now on.

The problem of deploying multiple robots and covering an environment can be formulated as the optimization of the aggregate objective function $\mathcal{H}(\mathbf{P})$. The objective function is defined over the free space $\Omega \subset \mathbb{R}^N$ and depends on the robot positions $\mathbf{P} \in \Omega^n$, with a total of n robots r_i in the network.

The domain Ω is assumed to be a convex polytope. Further, let $h: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ be a *performance function* to measure the degradation of the coverage performance with distance, e.g., the degrading performance of sensor measurements when distance increases, and $d: \Omega^2 \rightarrow \mathbb{R}_{\geq 0}$ be the function to measure distance between locations \mathbf{q} and robot positions \mathbf{p}_i in Ω . $d(\mathbf{q}, \mathbf{p}_i)$ may define a metric in Ω but that is no necessary requirement. The function h shall be strictly increasing over the image of d . Finally, given a (distribution) *density function* $\rho: \Omega \rightarrow \mathbb{R}_{\geq 0}$, the objective function is

$$\mathcal{H}(\mathbf{P}) = \int_{\Omega} \min_{i \in \{1, \dots, n\}} h(d(\mathbf{q}, \mathbf{p}_i)) \rho(\mathbf{q}) dF(\mathbf{q}) = E_{\Omega, \rho} \left[\min_{i \in \{1, \dots, n\}} h(d(\mathbf{q}, \mathbf{p}_i)) \right]. \quad (3.23)$$

$E_{\Omega, \rho}$ is the expected value of the degradation in coverage performance, which we aim to minimize.

One of the key concepts of the coordination method is that each robot r_i takes responsibility for a region Y_i , which we call the *region of dominance* of the robot. This results in a *tessellation* or *partition* of Ω , consisting of the collection of n regions, $\mathcal{Y} = \{Y_i\}_{i=1}^n$, with disjoint interiors and $\bigcup_{i=1}^n Y_i = \Omega$. The aggregate objective function can be rewritten as the *coverage functional* or *coverage cost*

$$\mathcal{H}(\mathbf{P}, \mathcal{Y}) = \sum_{i=1}^n \int_{Y_i} h(d(\mathbf{q}, \mathbf{p}_i)) \rho(\mathbf{q}) dF(\mathbf{q}). \quad (3.24)$$

The coverage cost of Equation (3.24) must be minimized with respect to the robot positions \mathbf{P} as well as the partition of space \mathcal{Y} .

Equation (3.23) and Equation (3.24) represent the classical *facility location problem* or *continuous p -median problem* (Okabe et al., 2000). As we have seen in Section 3.3.1, for fixed robot positions, the optimal partition \mathcal{Y} is the Voronoi tessellation. In similar fashion, we define the *generalized Voronoi tessellation* $\mathcal{V}(\mathcal{P}) = \{V_i\}_{i=1}^n$, with respect to the distance function $d(\mathbf{q}, \mathbf{p}_i)$ and generators located at the positions of the robots \mathbf{P} . The coverage cost becomes

$$\mathcal{H}_{\mathcal{V}}(\mathbf{P}) = \sum_{i=1}^n \int_{V_i} h(d(\mathbf{q}, \mathbf{p}_i)) \rho(\mathbf{q}) dF(\mathbf{q}), \quad (3.25)$$

and the Voronoi regions V_i are given as

$$V_i = \{\mathbf{q} \in \Omega \mid d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j), \forall j \in \{1, \dots, n\}, j \neq i\}. \quad (3.26)$$

A minimizer of $\mathcal{H}_{\mathcal{V}}(\mathbf{P})$ is found at the CVT, where each generator \mathbf{p}_i^* is located at the centroid \mathbf{c}_{V_i} of its Voronoi region V_i . For the general functions

h and d , a *generalized centroid* is computed as

$$\mathbf{c}_{V_i} = \mathbf{p}_i^* = \operatorname{argmin}_{\mathbf{p}_i \in V_i} \int_{V_i} h(d(\mathbf{q}, \mathbf{p}_i)) \rho(\mathbf{q}) dF(\mathbf{q}). \quad (3.27)$$

The coverage cost in Equation (3.25) is now minimized by continuously moving each of the robots r_i toward the generalized centroid \mathbf{c}_{V_i} of its Voronoi region V_i , while the partition $\mathcal{V}(\mathbf{P})$ is updated simultaneously. This complies with the theory on Voronoi tessellations introduced in Section 3.3.1. Note that the original Voronoi tessellation and CVT after Definition 3.1 and Definition 3.2 are recovered for the Euclidean distance function, $d(\mathbf{q}, \mathbf{p}_i) = \|\mathbf{q} - \mathbf{p}_i\|_2$. Figure 3.3 provides an illustration.

A robot r_i and a robot r_j are said to be *Voronoi neighbors* if their Voronoi regions V_i and V_j are adjacent. The neighborhood of a robot r_i is then specified by \mathcal{N}_i , the set of indices of all the $|\mathcal{N}_i|$ Voronoi neighbors r_j of r_i .

Now, we are interested in finding local minima of the coverage cost,

$$\min_{\mathbf{P}, \mathcal{V}} \mathcal{H}(\mathbf{P}, \mathcal{V}) = \min_{\mathbf{P}} \mathcal{H}_{\mathcal{V}}(\mathbf{P}, \mathcal{V}(\mathbf{P})) = \min_{\mathbf{P}} \mathcal{H}_{\mathcal{V}}(\mathbf{P}). \quad (3.28)$$

The Voronoi tessellation $\mathcal{V}(\mathcal{P})$ minimizes the cost for given \mathbf{P} and does no longer need to be included explicitly. As shown by Du et al. (1999) and Pimenta et al. (2008), a necessary condition in order to minimize $\mathcal{H}_{\mathcal{V}}(\mathbf{P})$ is

$$\nabla_{\mathbf{p}_i} \mathcal{H}_{\mathcal{V}}(\mathbf{P}) = \nabla_{\mathbf{p}_i} \hat{h}(\mathbf{p}_i, V_i) = \int_{V_i} \nabla_{\mathbf{p}_i} h(d(\mathbf{q}, \mathbf{p}_i)) \rho(\mathbf{q}) dF(\mathbf{q}) = \mathbf{0}^T. \quad (3.29)$$

Note that the computation of the cost $\hat{h}(\mathbf{p}_i, V_i)$ as well as the partial derivatives with respect to the position of the robot r_i , $\nabla_{\mathbf{p}_i} \mathcal{H}_{\mathcal{V}}(\mathbf{P})$ and $\nabla_{\mathbf{p}_i} \hat{h}(\mathbf{p}_i, V_i)$, respectively, only depend on the robot position \mathbf{p}_i and the positions of its Voronoi neighbors given by \mathcal{N}_i . The partial derivatives can thus be distributed, and one can say they are “*decentralized in the sense of Voronoi*” (Cortés et al., 2004). See also Figure 3.4 on the left for further description of the distributed computation of the CVT.

The configuration of a multi-robot system whose robots r_i act as generators, and have optimized the coverage cost of Equation (3.25) by converging to a final configuration that corresponds to a CVT, is called a *centroidal Voronoi configuration*. As discussed in Section 3.3.1 and shown in the works by Du et al. (1999), Cortés et al. (2004), and Pimenta et al. (2008), the Lloyd’s algorithm can be used for implementing such a behavior. The multi-robot system is typically modeled by the kinematic model of the single integrator after Equation (3.16) and the gradient descent controller of Equation (3.21)

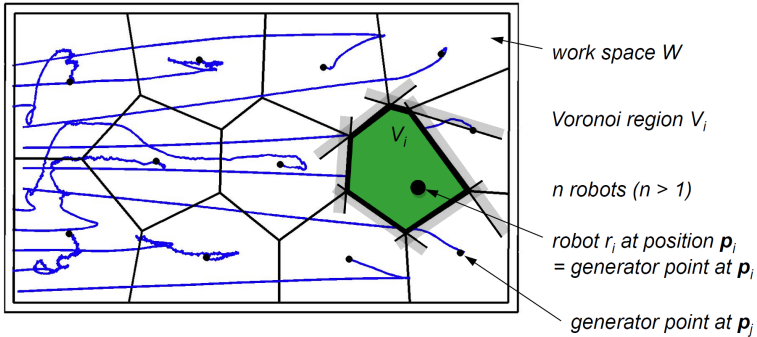


Figure 3.3: CVT and Voronoi coverage. A group of n robots r_i covers the bounded domain $\Omega \subset \mathbb{R}^2$, which is a subset of the workspace W . The robots act as generator points of the Voronoi tessellation; they deploy and optimize the coverage cost $\mathcal{H}_V(\mathbf{P})$, where the robot positions are given by $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n \in \Omega^n$.

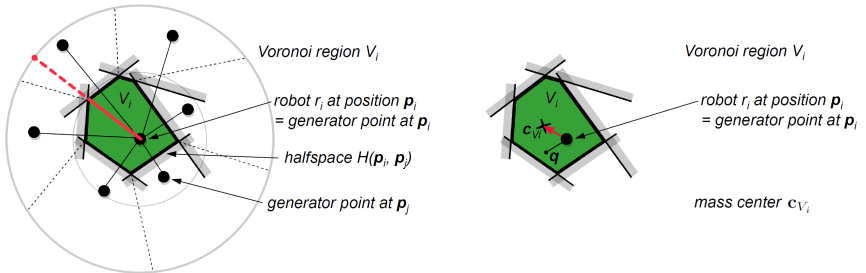


Figure 3.4: Distributed computation and gradient descent. Left: The computation of the CVT is “decentralized in the sense of Voronoi”, i.e., each Voronoi region only depends on the robot position \mathbf{p}_i and the positions of its Voronoi neighbors, which are contained within twice the maximum distance from the robot to a point at the boundary of the Voronoi region (solid red line). Right: The cost $\mathcal{H}_V(\mathbf{P})$ results from the integration of the distances from the robots at \mathbf{p}_i to each point \mathbf{q} in their Voronoi regions V_i . The Lloyd’s algorithm leads to a minimization of the cost and implements a gradient control law that moves each robot toward the mass center \mathbf{c}_{V_i} of its Voronoi region.

is a common choice for the control (see Figure 3.4, right). We would like to point out that the first-order dynamics of the model can generally be enforced by a low-level controller that makes the robots behave like single integrators.

The choices for functions h and d depend a lot on the problem at hand. For this thesis, the following choices are of relevance. We consider performance functions $h(d(\mathbf{q}, \mathbf{p}_i)) = d(\mathbf{q}, \mathbf{p}_i)^p$. For measuring distance, we look at functions $d(\mathbf{q}, \mathbf{p}_i) = \sum_{l=1}^m \|\mathbf{v}_l - \mathbf{v}_{l-1}\|_p$, i.e., the sum of distances along the sequence of segments $\mathcal{S}_{v_0, v_m} = \{\overline{\mathbf{v}_0 \mathbf{v}_1}, \overline{\mathbf{v}_1 \mathbf{v}_2}, \dots, \overline{\mathbf{v}_{m-1} \mathbf{v}_m}\}$, with $\mathbf{v}_0 = \mathbf{q}$ and $\mathbf{v}_m = \mathbf{p}_i$. $\|\mathbf{x}\|_p$ defines the p -norm. Two common setups are:

- *Setup 1: Euclidean distance.* For $p = 2$ and $m = 1$, we get $h(d(\mathbf{q}, \mathbf{p}_i)) = d(\mathbf{q}, \mathbf{p}_i)^2$ and $d(\mathbf{q}, \mathbf{p}_i) = \|\mathbf{p}_i - \mathbf{q}\|_2$. The Euclidean distance is a standard choice for open space and convex domains.
- *Setup 2: shortest path distance.* For arbitrary p , we get the length of the shortest path from \mathbf{q} to \mathbf{p}_i defined over m segments, such that the entire path is contained in Ω . For infinitesimal segment lengths, the shortest path approximates the shortest geodesic. The shortest path distance is used in the context of curved spaces and nonconvexities.

Let us in the following consider the specific case of setup 1 with $h(d(\mathbf{q}, \mathbf{p}_i)) = d(\mathbf{q}, \mathbf{p}_i)^2$ in more detail. The coverage cost of Equation (3.7) for $\mathcal{Y} = \mathcal{V}$ and Equation (3.25) become equivalent and each of the summands can be reinterpreted as the *polar moment of inertia* of the Voronoi region V_i at position \mathbf{p}_i ,

$$\begin{aligned} J_{V_i, \mathbf{p}_i} &= \int_{V_i} \|\mathbf{p}_i - \mathbf{q}\|_2^2 \rho(\mathbf{q}) dF(\mathbf{q}), \\ &= J_{V_i, \mathbf{c}_{V_i}} + M_{V_i} \|\mathbf{p}_i - \mathbf{c}_{V_i}\|_2^2. \end{aligned} \quad (3.30)$$

Here, the last equality follows directly from the *parallel axis theorem* known from mechanics, and $J_{V_i, \mathbf{c}_{V_i}}$ defines the polar moment of inertia of the Voronoi region V_i with respect to its centroid \mathbf{c}_{V_i} . The parallel axis theorem after Huygens-Steiner is stated in the book by Semat and Katz (1958) as follows.

Theorem 3.1. (Parallel Axis Theorem) *If the moment of inertia of a body about an axis through its center of mass is known, the moment of inertia of the body about any axis parallel to the first is given by the moment of inertia about the axis through the center of mass plus the product of the mass of the body by the square of the perpendicular distance between the two axes.*

Using Equation (3.25) and Equation (3.30), the coverage cost can be expressed by

$$\mathcal{H}_{\mathcal{V}}(\mathbf{P}) = \sum_{i=1}^n J_{V_i, \mathbf{c}_{V_i}} + \sum_{i=1}^n M_{V_i} \|\mathbf{p}_i - \mathbf{c}_{V_i}\|_2^2, \quad (3.31)$$

and the partial derivatives are calculated as

$$\nabla_{\mathbf{p}_i} \mathcal{H}_V(\mathbf{P})^\top = \nabla_{\mathbf{p}_i} f(\mathbf{p}_i, V_i)^\top = -2 M_{V_i} (\mathbf{c}_{V_i} - \mathbf{p}_i). \quad (3.32)$$

From this equation together with the condition $\nabla_{\mathbf{p}_i} \mathcal{H}_V(\mathbf{P}) = \mathbf{0}^\top$ we find again that the centroids \mathbf{c}_{V_i} are critical points \mathbf{p}_i^* of $\mathcal{H}_V(\mathbf{P})$. Moreover, it becomes apparent that Equation (3.32) indeed corresponds to the gradient descent method from optimization. The control law can be implemented accordingly as the linear proportional gradient descent controller given by Equation (3.21),

$$\mathbf{u}_i = -k_{1,i} \nabla_{\mathbf{p}_i} f(\mathbf{p}_i, V_i)^\top = k_{2,i} (\mathbf{c}_{V_i} - \mathbf{p}_i), \quad (3.33)$$

with positive control gains $k_{1,i}$ and $k_{2,i}$.

Finally, we turn toward the analysis of the convergence of the Lloyd's algorithm. Cortés et al. (2004) distinguishes the *continuous-time* from the *discrete-time* Lloyd's algorithm. Lloyd's algorithm, as it is introduced in Section 3.3.1 and originally formulated by Lloyd (1982), represents the discrete-time version. A continuous-time version, where both robot positions as well as partitions change in continuous time, is particularly relevant for robotics. The following two propositions are taken and adjusted from Cortés et al. (2004). They show the algorithm's completeness and asymptotic convergence to a local minimum.

Proposition 3.2. (Convergence of Discrete-Time Lloyd's Algorithm)

Let L be a continuous mapping $L: \Omega^n \rightarrow \Omega^n$ with the following properties:

- (1) $\|L_i(\mathbf{P}) - \mathbf{c}_{V_i}\| \leq \|\mathbf{p}_i - \mathbf{c}_{V_i}\|$, $\forall i \in \{1, \dots, n\}$ and L_i the i th-component of L , i.e., the distance to a centroid is not increasing.
- (2) If $\mathbf{p}_i \neq \mathbf{c}_{V_i}$, $\forall i \in \{1, \dots, n\}$, then there exists a j , such that $\|L_j(\mathbf{P}) - \mathbf{c}_{V_j}\| < \|\mathbf{p}_j - \mathbf{c}_{V_j}\|$, i.e., at least one robot moves toward its centroid at each iteration.

With \mathbf{P}^0 the vector of initial robot positions, the sequence $\{L^k(\mathbf{P}^0) \mid k \in \mathbb{N}\}$ converges to the set of centroidal Voronoi configurations. If this set is finite, then the sequence $\{L^k(\mathbf{P}^0) \mid k \in \mathbb{N}\}$ converges to a single centroidal Voronoi configuration.

Proof. Convergence follows from the proofs by Cortés et al. (2004) and Du et al. (1999). \square

Proposition 3.3. (Convergence of Continuous-Time Lloyd’s Algorithm)

Under feedback control, the robot positions $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n$ converge asymptotically to the set of critical points $[\mathbf{p}_i^]_{i=1}^n$ of the coverage cost $\mathcal{H}_V(\mathbf{P})$, i.e., the set of centroidal Voronoi configurations on Ω . If this set is finite, the robot positions converge to a single centroidal Voronoi configuration.*

Proof. Convergence follows from the proof by Cortés et al. (2004). □

3.6 Summary

This chapter presents the mathematical background and theoretical concepts which are central to this thesis. The used notation and terminology is defined. We provide basics in differential geometry, which are used to describe the geometry of curved surfaces. Voronoi tessellations from computational geometry are introduced. The tensor voting framework is presented as method for structure inference from point cloud data. Furthermore, concepts related to control theory, such as system models, the planning of feasible and optimal paths, as well as the Voronoi coverage method for the control of multiple robots, are described.

Chapter 4

The Concept of Hybrid Coverage

Coverage of workspace is an elementary task, which arises in multi-robot systems. Multi-robot coverage methods describe how multiple robots coordinate and partition their work load and workspace among each other. Depending on the tasks, which a group of robots has to accomplish, robot coverage can have different meaning: coverage equally involves static and dynamic robot distributions, and notions of sensing as well as actuation. Robot locations in mobile sensor networks for instance are optimized to provide good communication and sensor coverage of the environment (Bullo et al., 2009). Similar approaches lead to robot deployments that guarantee fair distribution of work loads among robots or short response times when providing services to allocated sites (Pavone et al., 2009). Another class of coverage tasks requires more permanent movements. In order to continuously monitor an environment, to inspect complex structures, or to apply tools to a surface in manufacturing, the robots perform coverage by sweeping through their workspaces (Choset, 2001).

In view of these different notions of coverage, we formulate the concept of *hybrid coverage*. The hybrid coverage concept will provide us with a basic formulation for the coverage by multiple robots throughout this thesis. Hybrid coverage combines deployment and sweeping motion by performing one after the other. Depending on the particular setup and algorithms used, either deployment follows sweeping, sweeping follows deployment, or both are interleaved.

The hybrid coverage concept is motivated by the multi-robot inspection task, which is to search for cracks in industrial environments with groups of

climbing robots, as outlined in the introductory chapters above. The central objective is to find weak points in the structure of an installation. In order to guarantee that a structure is safe and free of defects, a search must allow for complete coverage. Coverage in this context may just require to monitor an area from a certain distance by means of an imaging sensor but can also include visiting every surface location with a contact sensor. The use of multiple climbing robots and moving sensor probes along a surface eventually leads to deployment and sweeping. The concept of hybrid coverage looks at ways to combine deployment and sweeping with respect to the multi-robot inspection task in particular. However, the hybrid coverage concept is flexible enough, and, as will be shown below, can also describe tasks of multi-robot systems that are different from inspection.

We make use of the hybrid coverage concept to develop hybrid coverage solutions; such methods may build on and combine well-known coverage concepts from robotics literature. Moreover, the concept of hybrid coverage helps us to (re)think about coverage problems in a more unified way.

The hybrid coverage concept has first been mentioned at the International Conference on Autonomous Agents and Multiagent Systems (Breitenmoser et al., 2010c). Our work presented at the International Symposium on Distributed Autonomous Robotic Systems describes a specific implementation under the hybrid coverage concept for the inspection of curved surfaces (Breitenmoser et al., 2012). Related concepts are found again in our work on multi-robot pattern formation for robotic image and animation displays (Alonso-Mora et al., 2011, 2012b,c; Hauri et al., 2012).

We start in Section 4.1 with an overview on related coverage methods, which are useful components for the development of hybrid coverage methods. A definition of the hybrid coverage concept in Section 4.2 is followed by several exemplary variants of hybrid coverage in Section 4.3. Some of these variants are further evaluated by simulations and discussed with regard to their application to the MagneBike robots and the inspection scenario in Section 4.4. The summary in Section 4.5 concludes the chapter.

4.1 Related Work

Gage (1992) distinguishes three coverage behaviors in his considerations on how to command and control a many-robot system for the use in military applications: *barrier*, *sweep* and *blanket coverage*. Barrier coverage has the overall objective of clearance, and can be thought of, e.g., in a context of pursuit-evasion, as arranging robots on a barrier, such that the probability of penetration is minimized. Sweep coverage aims at moving multiple robots

across an area, such that the number of detections of targets per time is maximized and the number of missed detections per area is minimized. Finally, blanket coverage deals with the deployment of a group of robots into a static final robot configuration over an area, such that the detection rate of events appearing in this area is maximized. Among the three, sweep and blanket coverage are particularly important for us, since they reflect two substantially different behaviors¹. Whereas blanket coverage focuses on the spatial deployment of a group of robots, sweep coverage involves the dynamic covering of space, where the robots visit varying locations for servicing. Gage (1992) further recognizes the importance of commanding a group of robots as a whole, which is also relevant for coverage; abstractions on groups have been studied by Belta and Kumar (2004) and Ayanian and Kumar (2010), among others.

A prominent example for blanket coverage is the deployment of a mobile sensor network (Cortés et al., 2004), as outlined in Section 3.5.3. Closely related are techniques for surveillance and monitoring (Ganguli et al., 2007), or environmental monitoring and sampling (Schwager et al., 2006) with networked robots, which originate from applications in static wireless sensor networks. Examples for sweep coverage, or so-called *coverage path planning*, are reviewed in the survey by Choset (2001), and find applications in cleaning and painting, mowing and harvesting in agriculture, or industrial inspection and humanitarian demining.

Choset and Pignon (1997), Atkar et al. (2001, 2005) and Rekleitis et al. (2004) introduce coverage algorithms that work for single and multiple robots as well as in planar environments and on surfaces embedded in 3D space. These coverage algorithms are based on the *Boustrophedon decomposition*. The robots' free space is decomposed into cells which can be covered with simple back-and-forth sweeping motions. The cells are created by sweeping a (virtual) vertical line through the environment; when the line is broken due to obstacles, a *critical point* is generated and new cells are formed. The robots then visit one cell after the other, and coverage of all the cells finally results in complete coverage of the free space. An alternative is to construct a *covering spanning tree* over an area (Agmon et al., 2006). Online and offline algorithms for single and multiple robots exist for the tree generation. By moving along the spanning tree, the robots cover the area, which finally results again in complete coverage.

¹According to Gage (1992), formation control, deployment and recovery are seen as independent behaviors besides coverage; in our interpretation of the concepts proposed by Gage (1992), instead, we view coverage as a superior task, which may be composed in parts of deployment or formation subtasks.

Completely different approaches, which lead to alternative solutions to coverage problems, are found in *swarm-based methods*, which use either bio-inspired techniques like flocking and foraging, or physics-inspired techniques like particle systems. Among the methods for modeling flocks, we would like to mention the fundamental work by Reynolds (1987), and its reinterpretation under control theoretical aspects by Olfati-Saber (2006). Both describe flocking by the three fundamental steering behaviors of *coherence*, *separation* and *alignment*. Wagner et al. (1999) considers coverage of terrain by robots which leave traces that evaporate over time, mimicking stigmergy, i.e., the coordination by pheromones in nature. Another approach to swarm-based solutions results from so-called *physicomimetics* (Spears et al., 2006); robots are modeled and controlled as particles in a fluid or gas, following kinetic theory. The robots move dynamically in space and coverage conforms to stochastic predictions.

4.2 Hybrid Coverage Solutions

The hybrid coverage concept approaches the multi-robot coverage task by combining deployment with sweeping motion, or in other words, by the combination of blanket and sweep coverage. A hybrid coverage solution consists of two stages and is realized in one of the following two ways.

Definition 4.1. *Hybrid coverage of the first type (type 1)* conceptualizes a coverage method where, in the first stage, the robots spread out locally to cover an area while interaction keeps them in formation. In the second stage, the entire robot formation moves along distinct regions of the environment on the high level.

Definition 4.2. *Hybrid coverage of the second type (type 2)* conceptualizes a coverage method where, in the first stage, the robots deploy cooperatively within communication range and assign regions of operation to each other, which results in a partition of the given area. In the second stage, each of the robots takes care of its assigned region and sweeps over it locally.

The basic concept of hybrid coverage is described by two states, each representing one of the two stages, and three loops, as shown in Figure 4.1 and Algorithm 2. Two of the loops, one per stage, keep the algorithm in the current state, whereas the third loop represents a switch between the two stages, which results in the transition from one state to the other. Figure 4.2 illustrates both the hybrid coverage of the first type and the hybrid coverage of the second type.

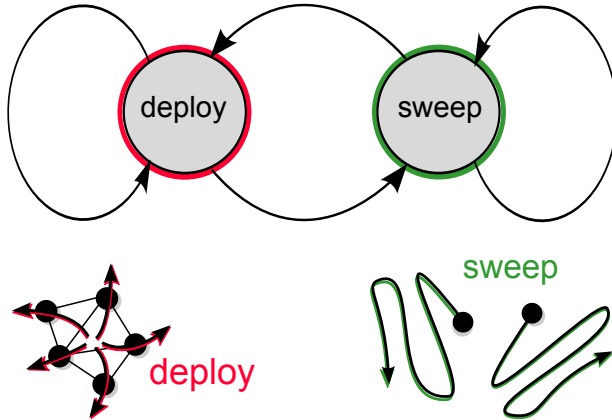


Figure 4.1: Concept of hybrid coverage. Top: Hybrid system for coverage, based on the two states of deployment and sweeping motion. Bottom: Deployment motion or blanket coverage (left) and sweeping motion or sweep coverage (right) by multiple robots (black disks).

Algorithm 2 Hybrid Coverage Algorithm

Require: Set of n robots, provided with modules for sensing, localization, environment modeling, communication and planning, as specified in Section 3.1. Subroutines that implement the two stages of deployment and sweeping for the required type of hybrid coverage.

```

1: loop {Coverage Main Loop}
2:   if (state == DEPLOY) then
3:     Deploy(type) // “blanket coverage”
4:   end if
5:   if (state == SWEEP) then
6:     Sweep(type) // “sweep coverage”
7:   end if
8: end loop

```

4.3 Variants of Hybrid Coverage

Next, we present several variants of the hybrid coverage concept, both for type 1 as well as type 2. Our goal in the following is to provide some demonstrative examples, which explain the rather general and abstract hybrid coverage formulation given in the definitions of the previous section.

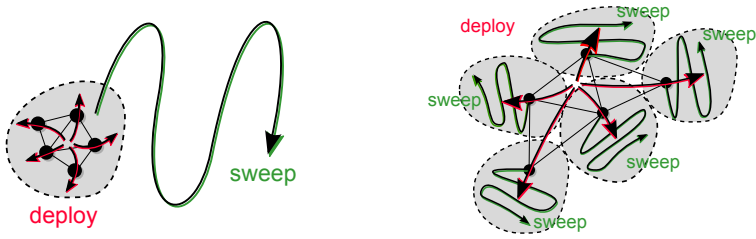


Figure 4.2: Two types of hybrid coverage. Left: Hybrid coverage of the first type (type 1). Right: Hybrid coverage of the second type (type 2).

4.3.1 Examples for Hybrid Coverage of Type 1

Concerning the hybrid coverage concept of type 1, the combination of multiple robots into groups, i.e., the abstractions on groups of robots, plays a central role (Ayanian and Kumar, 2010; Belta and Kumar, 2004). While the individual robots spread out and sweep a local area, the robots coordinate their motion and stay in a group. The group of robots is directed on the high level along a sweeping pattern through the environment. Thereby, the group of robots can be organized by general methods of multi-robot formation control. This leads to a tight coupling and coordination of the robots within the group. In the following two examples, we use flocking behaviors and elements from Voronoi coverage together with the abstraction on groups.

Coverage by Flocking and Sweeping

This first example shows a straightforward realization of the hybrid coverage concept of type 1 (see Figure 4.2 on the left, and Figure 4.4 for the realization of the concept). The coverage solution builds on a reactive approach, which is inspired by Koenig and Liu (2001) and is in line with the steering behaviors of Reynolds (1987); alternatively, the steering behaviors could be implemented by the consensus-based approach suggested by Olfati-Saber (2006).

Multiple robots are required to search an area as a group. The group of robots is controlled at high level by a *main sweep direction*, which forces the robots to move in the same direction and follow a common sweeping path. The individual robots map locations which they have already covered, using a grid representation. They scan the grid map continuously in order to determine their optimal moving direction. Each robot calculates the number of uncovered cells in each direction, 360° around its current position, within

a maximum scan radius². This gives a basic direction count, which is stored in an angular histogram (see also Figure 4.4, right). Typically, uncovered cells near the robot get higher priority and are counted several times (this is a first contribution of the enhancement rule, one of the used behavioral rules described in the following). Accordingly, a robot always tries to move over uncovered cells to reduce redundant coverage. A set of simple behavioral rules describes the utility of each direction for the robot. The following set of behavioral rules is used in our solution:

- *Attraction*. The attraction rule accounts for cohesion. The robots must stay in a group in order to maintain communication. Therefore, single robots which are positioned farther away than a critical distance from the group's center are attracted back to the center. In addition, directions pointing away from the center are valued lower than directions toward the center.
- *Repulsion*. The repulsion rule achieves separation. In order to avoid collisions, directions that lead to collisions are devalued.
- *Enhancement*. The enhancement rule includes alignment. Changing the moving direction is costly in general. Directions close to the current moving direction are weighted higher and a robot's need for frequent turning is decreased. In addition, directions aligned with the main sweep direction are enhanced.

According to these behavioral rules, the direction count is weighted. For each rule, the directions in the angular histogram are scaled by multiplying with a Gaussian distribution. The distribution is centered around a desired moving direction, which represents the behavioral rule. The distribution's standard deviation encodes the strengths of the weighting factors into the different directions. After the direction count has been calculated and scaled by all the behavioral rules in sequence, the robots choose their final moving direction to be the direction with the highest resulting direction count.

The main sweep direction can either be given externally from a higher-level controller or human input, or is determined by the robots themselves. For a regular update of the main sweep direction, the robots record the uncovered cells in every direction over a certain time period, and finally agree on a new main sweep direction. The new main sweep direction is the direction with most uncovered cells, resulting from the cumulated direction counts of all the interacting robots in the group.

²Note that the robots must consider the number of times a cell in their map has been covered, and not only count the number of uncovered cells, if the complete coverage of an area needs to be guaranteed.

Shape- and Goal-Based Pattern Formation

Pattern formation is—even though less apparent—another example that can be viewed as a variant of the hybrid coverage method of type 1. Let us assume a group of robots is locally distributed to represent a given pattern, such as an input image in our case. The input image is preprocessed by segmentation and the image areas that need to be represented by the robots are selected. The representation of a selected image area with a group of robots can be reinterpreted as a multi-robot coverage problem: multiple robots are to be deployed locally in order to cover the projection of the selected image area in their workspace. This relates to the first stage of the hybrid coverage concept. Now, if we are furthermore interested in moving patterns, such as image areas evolving over a sequence of input images in an animation or video, the entire robot formation will move over the workspace in order to represent a moving pattern. And we notice that this just complies with the second stage of the hybrid coverage concept.

In the following, we present a brief overview of our *shape-based* as well as *goal-based* pattern formation algorithms for animation display. As pattern formation is not the topic of this thesis, however, we will not go into all the details. For a thorough discussion of the topic, the reader is referred to Hauri et al. (2012) regarding shape-based pattern formation and to Alonso-Mora et al. (2011, 2012b,c) regarding goal-based pattern formation. We here limit ourselves to a consideration of the algorithms in light of the hybrid coverage concept.

The shape-based pattern formation algorithm again relies on flocking after Reynolds (1987) but is now implemented by the consensus-based approach of Olfati-Saber (2006). The original steering behaviors are complemented with an additional shape-steering rule that constantly drives the robots toward the interior of a selected image area. Robots outside the image area are attracted toward the area and robots located on the contour of the image area are pulled inside. Robots that reside in the interior of the image area are no longer influenced by the shape-steering rule. They deploy locally under the original steering behaviors of coherence, separation and alignment, and eventually adopt the shape of the image area. The algorithm makes use of different types of α -, β - and γ -agents to model the steering behaviors (Olfati-Saber, 2006). For each robot, a disk which is centered at the current robot position is intersected with the image area. The shape-steering rule is then implemented by additional γ -agents, which appear in one of two configurations: a robot on the outside is attracted by a γ -agent located on the contour within shortest Euclidean distance, whereas a robot in the inside is attracted

by a γ -agent which is positioned at the centroid of the area that results from the intersection of the disk with the image area.

The image area may be transformed from frame to frame in the animation, and thus the robot formation is dragged along by the moving shape, which realizes shaped flocking.

An alternative approach is to explicitly determine the goal positions of individual robots within the selected image area, which leads to a goal-based representation of the image. The goal positions are distributed over the selected image area by computing a CVT, which bears similarities to Voronoi coverage. At each frame, the set of goal positions is updated and reassigned to the set of robots using an auction algorithm. The robots move to their assigned goal positions while avoiding collisions locally, which results in a robot formation. The selected image area may again be moved through a sequence of input images. The goal positions need to be updated in this case. An estimate for the translation of the goal positions is obtained from the translation of the centroid of the image area between two subsequent frames. The rotation of the goal positions is estimated by matching the contour of the current with the previous image area. From these estimates, the new goal positions can be initialized, and they finally result from the computation of the CVT. The initialization reduces computation time and disparities between subsequent goal sets. The robots track the new goal positions and the formation moves again along a sweeping path at the high level.

4.3.2 Examples for Hybrid Coverage of Type 2

The hybrid coverage concept of type 2 is essentially based on the classical principles of cell decomposition (Choset and Pignon, 1997; LaValle, 2006). The decomposition of an environment may depend on the environment geometry, on the robot configuration, or on both of them. A group of robots deploys and collaboratively partitions the environment. Once the environment is decomposed, the robots allocate regions of the environment among each other. The allocation process can be implemented by standard task assignment methods. After decomposition, each robot starts to cover its assigned region. If the regions need to be swept completely, the robots generate sweeping patterns or space filling curves within their regions. The cell decomposition of the first stage leads to loose robot coordination. In the second stage, the robots may fulfill their coverage tasks independently, being only loosely coupled, or even decoupled for limited time periods.

Coverage by a Robotic Band

This is a first example for a hybrid coverage method of type 2. The coverage solution generates an exact cellular decomposition similar to the approach of Rekleitis et al. (2004) and achieves spatial coverage according to Spears et al. (2006). At start, a group of robots forms a robotic band (see Figure 4.5). Two robots limit the operation area of the entire band on the left and on the right side perpendicular to a main sweep direction. If the operation area is bounded, the two robots follow the boundary of the environment. All the other robots line up in-between. The main sweep direction is either preset or determined by the robots collectively. The robots move in parallel formation and deploy on the search for critical points in the main sweep direction. Critical points define boundaries of adjacent cells in the decomposition. Linked to each critical point is a cell as well as a new open task to accomplish (e.g., a coverage task). When a critical point is detected by the robotic band, one of the robots from inside the band is assigned to the cell and left behind to sweep the cell. By leaving robots behind for coverage, as the robotic band advances, a robot network is established across the area, over which information can be communicated among the deployed robots. In particular, this network is used to assign open tasks to idle robots.

We distinguish five different types of critical points:

- *START critical point.* The **START** critical point is set at the initial position of the robots.
- *IN critical point.* The **IN** critical point is instantiated when the robotic band is separated by an obstacle and line-of-sight between robots in the band is lost.
- *MIDDLE critical point.* The **MIDDLE** critical point occurs (1) when the distance to the previous critical point exceeds a given maximum distance, or (2) when the spanned cell area becomes equal in size to the area of previously created cells. In the first case, the robots retain the possibility of relative localization among each other and prevent loss of communication. In the second case, the robots realize equal cell sizes for a balanced partition of the environment.
- *OUT critical point.* The **OUT** critical point terminates an obstacle. It is set by a robot after following an obstacle boundary and suddenly finding itself moving against the main sweep direction.

- *END critical point.* The END critical point results when two robots follow the boundary of the environment or the specified operation area, and finally meet each other.

For the representation of the critical points and the cells through which they are connected, we use a so-called *Reeb graph* (Rekleitis et al., 2004). The Reeb graph $G_R = \{V_R, E_R\}$ keeps track of the set of critical points, which are the graph's vertices $v \in V_R$, and the cells connecting them, which are the graph's edges $e \in E_R$. The Reeb graph additionally stores information on the status of the cells, e.g., if a cell has been covered. Each robot constructs its own Reeb graph of the environment and updates the Reeb graph as soon as a new critical point is detected.

There are many possibilities of how to handle critical points and their related tasks. For example, if an IN critical point is detected shortly after a MIDDLE critical point, a cell with small area is created. The robotic band may not leave a robot behind for covering the cell but instead adds the coverage task to a task protocol. As soon as a previously deployed robot has completely covered its cell, it inherits the task of covering this nearby cell with small area.

If an IN critical point is detected at an obstacle, the robotic band separates into two groups, which both continue the deployment, one on each side of the obstacle. At the point where only two robots are left, as all the other robots have already been deployed, the robots issue a new open task for the one side of the obstacle and continue deploying on the other side. The remaining two robots move forward until three additional critical points have been created. The first two critical points indicate the robots' upcoming coverage work and the third critical point serves as a starting point for a new task, at which robots can gather after completion of their previous tasks.

Hybrid Voronoi Coverage

In the last example, we present yet another method that corresponds to the hybrid coverage concept of type 2. The coverage solution combines Voronoi coverage, as described in Section 3.5.3, with subsequent sweeping of the constructed Voronoi regions. It is similar to the previous example—just that now the cell decomposition depends on the robot positions rather than on the positions of detected critical points along the obstacle or environment boundaries. The method first spreads the robots into disjoint regions of dominance, applying the Voronoi coverage method. Once the robots are deployed, each robot takes care of its assigned Voronoi region and sweeps the region along a covering path. Here, basically any space filling curve can be used, as long as

the kinematic constraints of the robots are met. Examples are swaths similar to those used in the Boustrophedon decomposition on planar and curved surfaces (Atkar et al., 2001; Choset and Pignon, 1997).

Related concepts are found in multi-robot exploration (Haumann et al., 2011; Solanas and Garcia, 2004). A CVT is computed to partition an unknown environment and deploy one robot per region in a first stage. Each robot becomes responsible for the exploration of its assigned region in a second stage. The Voronoi regions can be static or may change dynamically as the robots move through the environment.

4.3.3 Combination of Type 1 and Type 2

Moreover, hybrid coverage of type 1 and type 2 can be combined, which leads to countless further variations. Typical combinations of hybrid coverage methods can be obtained by varying the scope and involving different hierarchical levels. Figure 4.3 shows three examples.

In the first example, a hybrid coverage method of type 2 is combined with a hybrid coverage method of type 1 on the low level by grouping several robots together, which then sweep a decomposed cell (Figure 4.3, top left). Second, a hybrid coverage method of type 2 is extended by hybrid coverage of type 1 on the high level. A group of robots deploys and covers a part of an area under the hybrid coverage method of type 2. The robots then relocate and redistribute to cover a next part of the area, again under the hybrid coverage method of type 2. By iterating the hybrid coverage of type 2, the group of robots moves along a sweeping path and in fact implements a hybrid coverage method of type 1 (Figure 4.3, top right; see also Section 6.4.2). Another example of hierarchical composition results from combining several hybrid coverage methods of type 2 at different levels (Figure 4.3, bottom center). A group of robots deploys into different regions, and in each region, a subgroup redeploys into subregions, which finally leads to a subdivision of space. Such hierarchical compositions also relate further to ideas of multi-resolution or iterative deepening.

4.4 Results

So far, several hybrid coverage solutions have been proposed and existing coverage methods have been reconstructed under the hybrid coverage concept. In this section, we further evaluate three variants of hybrid coverage from Section 4.3.1 and Section 4.3.2: “coverage by flocking and sweeping” (hereafter, denoted as variant 1), “coverage by a robotic band” (variant 2), and

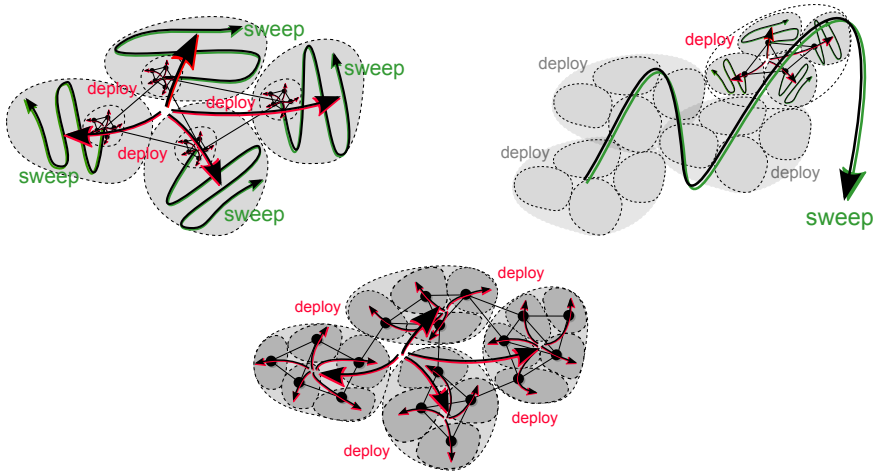


Figure 4.3: Combinations of hybrid coverage. Top left: Hybrid coverage of type 2 combined with hybrid coverage of type 1 on the low level. Top right: Hybrid coverage of type 2 combined with hybrid coverage of type 1 on the high level. Bottom center: Combining several hybrid coverage methods of type 2 at different levels.

“hybrid Voronoi coverage” (variant 3). We first present results from Matlab simulations, and then discuss how the algorithms apply to the inspection task and the MagneBike robots.

4.4.1 Comparison of Hybrid Coverage Algorithms

We analyze the algorithms on the basis of two polygonal 2D environments with overall dimension of $3\text{ m} \times 5\text{ m}$: a free rectangular area (see Figure 4.4) and a nonconvex area with a single triangular obstacle in the center (see Figure 4.5). The number of robots is varied from 4 to 10. The robots move with a fixed speed of 5 cm/s and their sensor footprint covers a circle with radius 3 cm . The environments have a grid overlaid and the percentage of coverage is computed from the number of covered cells at a given point in time. The robots are assumed to be point robots; thus there is no strict collision avoidance included in the simulations³. The robots are modeled as single integrators and able to move in any direction. The simulations

³For implementations on real systems, local collision avoidance can either be embedded directly into the coverage algorithms or be conferred to a controller on a lower level.

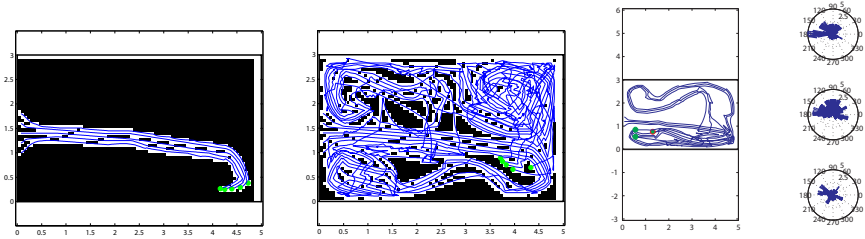


Figure 4.4: Coverage by flocking and sweeping. Coverage sequence of variant 1. Left and center: A group of robots covers the free rectangular area. Right: The angular histograms are shown for three robots while they are covering the area.

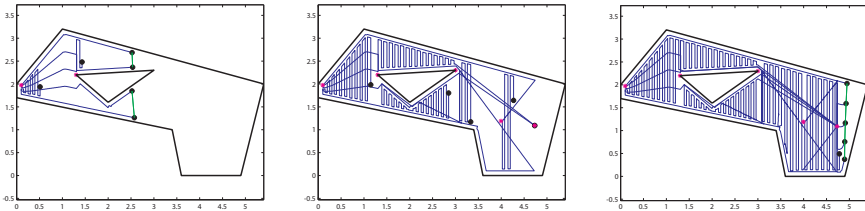


Figure 4.5: Coverage by a robotic band. Coverage sequence of variant 2. A group of robots covers the nonconvex area with a single triangular obstacle in the center.

are ideal in that no noise or physics, e.g., friction, are included and perfect localization is assumed. The focus of the evaluation is on the resulting overall robot behavior and the generated coverage paths.

A simple random coverage algorithm is included as an additional benchmark; each robot changes its direction at random whenever a boundary is reached or another robot comes too close. Matlab's uniformly distributed pseudorandom numbers are used for generating the new angle of a direction update. The result for the random coverage algorithm in Figure 4.7 represents the average value over ten simulation runs.

Variant 1 weights the direction count depending on several behavioral rules and thus needs a fair amount of tuning. The maximum distance a robot is allowed to move away from the group's center, all the Gaussian distributions for scaling the direction counts, as well as the update frequency of the main sweep direction must be adjusted. In contrast, variant 2 and variant 3 require less tuning. The sweeping pattern needs to be defined

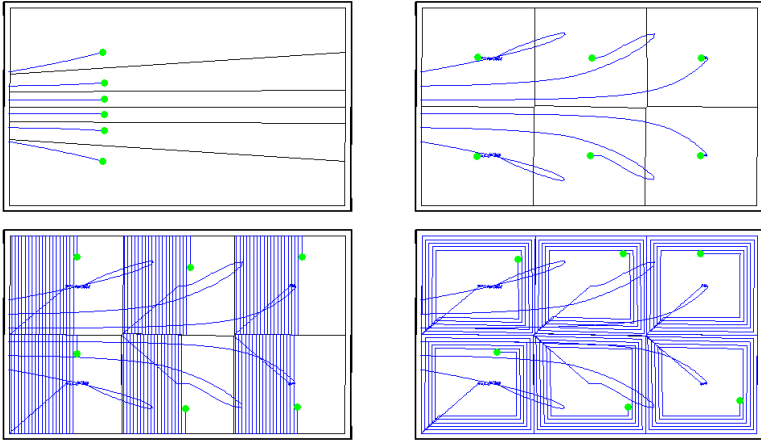


Figure 4.6: Hybrid Voronoi coverage. Coverage sequences of variant 3. A group of robots deploys (top) and covers the free rectangular area by using either back-and-forth (bottom left) or spiraling sweeping patterns (bottom right).

for both variants; the distance between back-and-forth motions is set to a bit less than twice the robot’s sensor radius. For variant 2, we need to specify the maximal cell size. In the simulations, we set the maximum cell size to the optimal value with respect to the environment and number of robots. Variant 3 requires a choice of performance and distance functions, as well as of a density distribution; we use setup 1 from Section 3.5.3 and a uniform density in the simulations. Typical sequences from coverage runs with variant 1, variant 2 and variant 3 are shown in Figure 4.4, Figure 4.5 and Figure 4.6.

Simple Environments. Figure 4.7 presents the coverage over time, which results from simulation runs with the three coverage algorithms for the free rectangular area and varying group sizes. Clearly, we can say that a larger group of robots reaches a specific percentage of coverage faster compared to a smaller group. However, doubling the number of robots does not result in half the coverage time in general. Effects of redundant coverage and coordination overhead get noticeable.

All three variants outperform the random coverage algorithm; the random coverage algorithm generally requires more than twice as much time to reach 95 % of coverage. Probabilistic or random coverage algorithms become only competitive when the sensing tool is noisy, such that the increased redun-

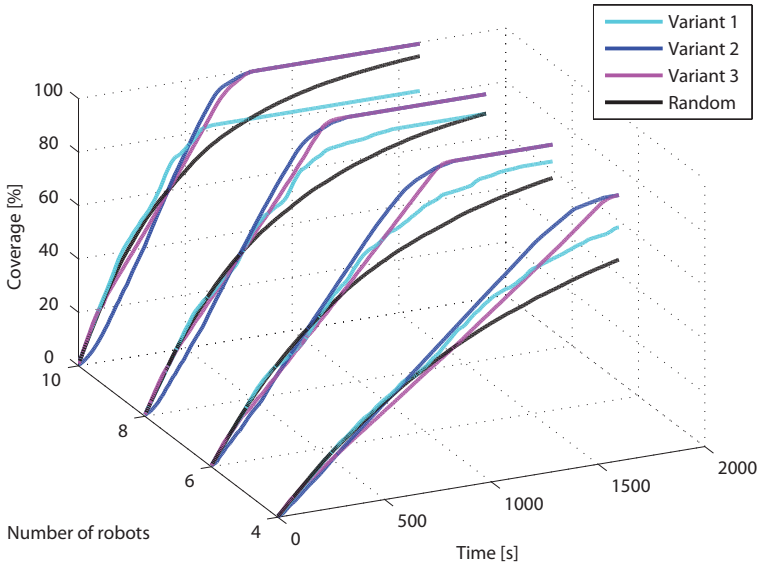


Figure 4.7: Comparison of hybrid coverage variants. Coverage over time for a group of 4, 6, 8 and 10 robots in the free rectangular area for variant 1, variant 2, variant 3 and the random coverage benchmark.

dancy in coverage resulting from the randomized solution can pay off (Gage, 1992; Pugh and Martinoli, 2007).

For this simulation, variant 1 uses a limited scan radius and only considers if a cell has been covered or not in the direction count, i.e., not counting the number of times a cell has been covered. Thus, as seen from Figure 4.7, complete coverage is not guaranteed; however, a final percentage of roughly 90 % of coverage is still reached in most cases. Note that the extension mentioned in the footnote of Section 4.3.1 can resolve this issue. Variant 1 slows down after around 50 % of the environment is covered, since it becomes more difficult to find uncovered cells and redundant coverage of cells is increasing. Variant 2 and variant 3 are slower at the beginning, since the robots first need to deploy; this in addition leads to redundant coverage. The methodical coverage of variant 2 and variant 3 finally pays off when the last 20 % of the area to cover are reached. Variant 2 and variant 3 perform equally well and achieve complete coverage of the free rectangular area.

More Complex Environments. Although variant 2 and variant 3 cover environments of simple geometries well, they highly depend on the complexity of the environment. Obstacles in the environment are not so much a problem for variant 1, however, unequal cell sizes and idle times result for variant 2. Moreover, nonconvex environments present a general challenge for variant 3, as Voronoi coverage has originally been designed for convex environments only. This is an issue which we will be concerned with in the upcoming chapters. As can be seen from Figure 4.5 on the left in the case of variant 2, obstacles introduce additional critical points, which leads to a higher number of cells and split up robot bands. More critical points, cells and split-up groups of robots mean an increased overall complexity of the system. Actions to take at a critical point depend on the critical point's type and the number of robots available. The risk of missing robots for fulfilling a task increases, and finally constrains robot subgroups to halt and wait for additional robots to help.

Sweeping Patterns. Variant 2 and variant 3 require the robots to sweep cells. Figure 4.6 shows several phases of variant 3 for six robots in the free rectangular area, once using back-and-forth motion and once spiraling motion for sweeping. As we assume constant speed in our simulation, the length of a robot's path grows linearly and is equal for both sweeping patterns. However, depending on the pattern, the accumulated turn angle and the percentage of coverage change over time. Turns are costly as a real robot has to slow down or consider them in its trajectory. Therefore it is important to adequately adjust cell shapes and available sweeping patterns under consideration of the environment.

Search for Defects. With regard to the inspection task, we further investigated the search for defects in the environment. In our simulation, ten line segments of varying length, which represent cracks, are randomly distributed over the area. As soon as a robot moves over such a crack, the defect is considered to be detected. We find that the random coverage algorithm locates the first defects faster than the three other coverage solutions, which is due to the fact that the random coverage algorithm distributes the robots faster over the whole environment. However, in each experiment the three coverage solutions find the last of the ten cracks faster than the random coverage algorithm. In an environment with many defects, a random sampling strategy might be superior; however, under the assumption that the occurrence of a defect is the exception, complete coverage is key to provide any guarantees.

4.4.2 Application of Hybrid Coverage to Inspection

Next, we discuss the three variants regarding their applicability to the inspection task and the MagneBike robots. As described in Section 2.2.1, the MagneBike robots have a size of $18.5 \text{ cm} \times 14.3 \text{ cm} \times 17.0 \text{ cm}$, and a typical NDT sensor probe has a diameter of around 3.0 cm . Therefore, due to the substantial difference in the size of MagneBike and the size of the sensor footprints, it is not possible to seamlessly cover the surface with all the robots in side by side formation. The coverage solution of variant 1 adjusts the distance between the robots by setting up attraction and repulsion appropriately, whereas variant 2 and variant 3 assign every robot its own region to provide for enough space. For all three variants it is possible that the robots always stay connected in a group; communication and relative localization between neighboring robots are enabled. The robots may localize themselves independently with respect to the workspace, or estimate their global positions through relative localization to their neighbors (see also Chapter 9).

Robustness and Efficiency. Variant 1 is robust to single robot failure; if a robot fails it is not included in the group anymore, and the remaining robots rearrange. However, variant 1 does not optimize travel distance. Generated paths traverse the environment repeatedly and from different directions and positions. Even though MagneBike is able to negotiate diverse geometries, certain maneuvers are more risky and time-consuming than others, and thus should be greatly avoided.

Variant 2 is prone to failures of single robots, since robots in the band formation depend on each other. Failure of a robot may lead to disconnection of a band. However, shorter paths compared to variant 1 are sufficient to deploy the robots in the environment. Especially structures with preferred directions, such as an axis of symmetry, or rotationally symmetric structures like cylindrical tubes, can be covered efficiently. This motivates an implementation of variant 2 on the MagneBike robots, and to use it for the coverage of the widespread symmetric and less complex industrial structures, with potentially a priori known geometry.

Variant 3 falls between the first two variants. It combines benefits from both sides, and applies more generally. Being mathematically well-defined, complete and robust but less dependent on the environment geometry, which is particularly useful for applications in a priori unknown environments, makes variant 3 a meaningful candidate for the potential application to an inspection task, as outlined in Chapter 1. However, issues in handling obstacles and curved surfaces exist, which need further study.

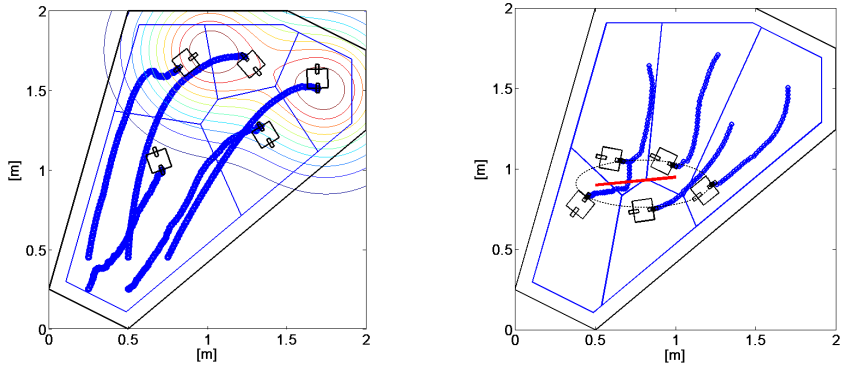


Figure 4.8: Expert knowledge and user guidance. Left: A dense robot configuration is induced around locations where a defect is expected. Right: Once a defect (red line) has been detected, the robots can be commanded to align for collaborative inspection or maintenance.

Expert Knowledge and User Guidance. The inspection task requires the MagneBike robots to search for defects in the surface of the industrial structures. For such a process, it is essential that robots are able to incorporate existing prior knowledge or expert knowledge. For instance, a human expert or experienced inspector may indicate critical areas of a structure beforehand, and the robots must intensify their investigation near those areas during inspection. Alternatively, an inspector may be given the possibility to assist and influence the group of robots in order to adjust the coverage process during runtime.

Variant 1 and variant 2 include a main sweep direction, which controls the overall motion of the group of robots. The main sweep direction enables an inspector to direct the robots to different areas of interest in an environment. In addition, the inspection of specific areas can be intensified by influencing the direction count of variant 1, or the cell size and lateral width of back-and-forth motions in variant 2.

Variant 3 features a similar functionality. The density function ρ of the Voronoi coverage method (refer to Section 3.5.3) can be used to gather robots at areas of special interest. During inspection, an inspector can increase the density values for the areas in the structure which are experienced to fail more likely. The Voronoi regions get smaller and the robots get more time to spend; a robot can inspect a critical area more thoroughly, or cover the area several times, which increases robustness through redundancy (see Figure 4.8, left).

Besides, the density function allows for formation control. Multiple robots can be controlled to form up along a defect for collaborative inspection or maintenance (see Figure 4.8, right).

4.5 Summary

In this chapter, we propose the concept of hybrid coverage. The concept combines methods from blanket and sweep coverage to design new hybrid coverage solutions. We show two types of realization, which differ in the order and scope deployment and sweeping motions are combined. The hybrid coverage concept furthermore offers an alternative perspective to look at coverage problems. Several variants of hybrid coverage have been demonstrated by examples, and three hybrid coverage methods were evaluated for their use in the inspection scenario.

A hybrid Voronoi coverage solution, which is a realization under type 2, showed particularly interesting properties with respect to inspection. The coverage solution deploys the robots by Voronoi coverage in the first stage. Each robot is assigned to one Voronoi region, which provides enough space for a single robot to maneuver. The robots sweep their assigned Voronoi regions in the second stage.

In addition, the solution allows for extensions by a hybrid coverage method of type 1, creating a hierarchical coverage solution. The solution can be extended on the lower level by grouping several robots together; instead of single robots, several groups of robots now deploy and each group operates in one of the Voronoi regions. Or, the solution is extended on the upper level by repeated execution; once the Voronoi regions are swept, the robots relocate and deploy again, which leads to incremental coverage of an area by a growing sequence of bounded Voronoi tessellations along a sweeping path.

However, there remains an open question: How can Voronoi coverage be adjusted to cope with more realistic environments? This includes the operation under constraints. For inspection tasks in geometrically complex environments, this includes concretely the operation under constraints imposed by the environment's geometry, such as nonconvexity and curved spaces. Voronoi coverage for such constrained environments will be our subject for the next two chapters.

Chapter 5

Multi-Robot Coverage under Constraints

In this chapter we show that complications occur when the original Voronoi coverage method is applied to deploy a group of robots in nonconvex environments. Many approaches in multi-robot coverage assume environments that are known, planar and convex, and use robots that are unconstrained. However, real-world applications impose various constraints on the robots. An environment might be unknown a priori and the geometry of the environment might be arbitrarily complex. Visibility of the robots' sensors might be restricted by occlusions. Moreover, the communication among robots and their energy budgets might be limited. These constraints need to be taken into consideration when designing algorithms for the control of multiple robots.

In our work, we focus on constraints given by the geometric complexity of an environment. Motivated by realistic applications, we study multi-robot coverage in known and unknown nonconvex environments, including areas with free-standing obstacles and areas with nonconvex boundaries. In such nonconvex environments, two points cannot be connected by a straight line that is fully contained in the environment anymore, which complicates the robot coverage. If an environment is also unknown, the environment needs to be explored by the robots before or during the coverage process.

In the following, we present a coverage solution formulated for *continuous space* that builds on the Lloyd's algorithm and a path planning algorithm. The two algorithms are executed consecutively on an upper and lower level in the hierarchical system. We implement a local path planner using the TangentBug algorithm. The coverage solution spreads the robots over the environment while taking care of the nonconvexities by the built-in obstacle

avoidance behavior. The robots explore obstacle boundaries and apply a goal projection procedure to constrain outlying goal positions back onto the accessible free space. The interaction between the two algorithms results in global convergence of the robots to a final constrained centroidal Voronoi configuration. In addition, an exploration solution related to Voronoi coverage is proposed, which controls multiple robots to explore an unknown nonconvex environment prior to coverage.

Portions of the research presented in this chapter appeared in the International Conferences on Robotics and Automation (Breitenmoser et al., 2010b; Haumann et al., 2011). Section 5.1 starts with a discussion of related work on Voronoi coverage that considers different types of constraints. In Section 5.2, an overview of the TangentBug algorithm is provided. Section 5.3 describes our nonconvex coverage algorithm and its properties. Then, Section 5.4 briefly explains the transformation to star-shaped domains, which offers an alternative to handle nonconvexities. The nonconvex coverage algorithm is tested in simulations and experiments with a group of e-puck robots in Section 5.5. The chapter concludes with a summary in Section 5.6.

5.1 Related Work

A multi-robot coverage approach that has attracted considerable attention in recent years is Voronoi coverage (Cortés et al., 2004). The reader refers to Chapter 3, Section 3.5.3, for an introduction to Voronoi coverage. We limit our literature review of coverage methods to Voronoi coverage methods and their modifications to satisfy additional constraints. Constraints are typically introduced when multiple objectives must be achieved.

The seminal work by Cortés et al. (2004) includes a demonstration of the Voronoi coverage method for robots with nonholonomic constraints, such as the unicycle model. Pimenta et al. (2008) shows how to constrain the Voronoi regions to deal with robots of finite size.

Another constraint is due to the lack of knowledge of the density function ρ in the Voronoi coverage method. Whereas Cortés et al. (2004) assumes that ρ is known, Schwager et al. (2006, 2009) iteratively estimates ρ from sensor measurements while adjusting the robot configuration accordingly. The methods presented by Cortés et al. (2005) introduce a limited sensor or communication range, which constrains each Voronoi region within a disk of fixed radius centered at the robot's position. In the work by Gusrialdi et al. (2009), the sensor range is constrained by the robot orientation according to an anisotropic sensor model.

Additional constraints may also aim at keeping the robots safe during the coverage procedure. The objectives of providing coverage and maintaining a desired energy level of the individual robots can be achieved by allowing the robots to return to a charging station when they run low on energy (Derenick et al., 2011). The work by Carpin (2012) tries to protect the robots by trading off coverage against covertness, i.e., minimizing exposure to multiple adversarial observers.

Objectives of different priorities can alternatively be formulated with respect to task space control. The method of Antonelli et al. (2011) handles free-standing obstacles in a task-priority fashion by giving obstacle avoidance priority over Voronoi coverage. An alternative method by Caicedo-Núñez and Žefran (2008a) transforms a nonconvex environment through a diffeomorphism to a corresponding convex environment, in which regular Voronoi coverage can be applied. The method is computationally expensive and leads to solutions that generally differ from the optimal coverage solutions in the original space. For convex environments with free-standing obstacles, some of these issues have been resolved (Caicedo-Núñez and Žefran, 2008b). The method of Pimenta et al. (2008) uses Voronoi coverage with the geodesic distance to achieve coverage of environments with nonconvex boundaries. Unfortunately, the exact computation of geodesic distances is computationally rather costly.

The Voronoi coverage methods may deploy robots in unknown environments but do not explore the environment intentionally. A popular exploration strategy is frontier-based exploration (Yamauchi, 1998). The method of Burgard et al. (2005) selects goal points on the frontier between explored and unexplored areas by simultaneously considering the utility of unexplored areas and the cost for reaching these areas. The work by Solanas and Garcia (2004) uses k-means clustering to ensure that multiple robots explore different regions of the environment. Similar to Voronoi coverage, Haumann et al. (2010) uses a Voronoi partition and optimizes an aggregate objective function to explore an unknown convex environment with a group of robots.

5.2 Preliminaries

As our coverage solution combines multi-robot coverage and path planning in a modular manner, there are various path planning algorithms which could potentially be used. Vector field histograms, artificial potential fields and the family of Bug algorithms are examples of sensor-based local path planners. In the following sections, we will present an implementation of the cover-

Algorithm 3 TangentBug Algorithm

Require: Robot r at start position $\mathbf{p} = \mathbf{p}^0$, provided with a range sensor, a local reduced visibility graph $G_{\text{vis}} = \{V, E\}$ and a goal position \mathbf{g} .

```

1: loop {Main loop}
2:   while no local minimum detected,  $\exists v \in V, d(\mathbf{v}, \mathbf{g}) < d(\mathbf{p}, \mathbf{g})$  do
3:     if goal reached then
4:       return
5:     end if
6:      $\{\mathbf{p}, G_{\text{vis}}\} \leftarrow \text{MoveTowardGoal}(\mathbf{p}, \mathbf{g}, G_{\text{vis}})$ 
7:   end while
8:   while leave condition invalid,  $d(\mathbf{v}, \mathbf{g}) \geq d_{\min}(\mathbf{g}), \forall v \in G_{\text{vis}}$  do
9:     if goal reached or loop around obstacle completed then
10:      return
11:    end if
12:     $\{\mathbf{p}, G_{\text{vis}}, d_{\min}(\mathbf{g})\} \leftarrow \text{FollowBoundary}(G_{\text{vis}})$ 
13:  end while
14:  while  $d(\mathbf{p}, \mathbf{g}) \geq d_{\min}(\mathbf{g})$  do
15:     $\{\mathbf{p}, G_{\text{vis}}\} \leftarrow \text{PerformTransitionPhase}()$ 
16:  end while
17: end loop

```

age solution that uses the TangentBug algorithm; therefore we include some background information on the TangentBug algorithm in this section.

The *TangentBug algorithm* is a derivative of the family of Bug algorithms (LaValle, 2006), and was originally introduced by Kamon et al. (1995, 1998). TangentBug is a simple but efficient sensor-based planner, capable of handling unknown environments by using a range sensor. The range can assume any value, from zero, i.e., the range sensor is reduced to a contact sensor, to infinity, i.e., the entire visible domain can be seen at once. The length of the robot's path usually decreases with increasing range of the sensor. TangentBug shows the two behaviors which are characteristic for Bug algorithms: *motion-toward-goal* and *boundary-following*. A description of the TangentBug algorithm is given in Algorithm 3 according to Kamon et al. (1998). In every iteration step of the algorithm, the robot constructs a local version $G_{\text{vis}} = \{V, E\}$ of the *reduced visibility graph* (LaValle, 2006), as if the robot's local range information represented all the obstacles in the environment. During the motion-toward-goal behavior, the robot moves in the locally optimal direction along the shortest path in G_{vis} . The function

$d(\mathbf{v}, \mathbf{g})$ measures the Euclidean distance of a vertex \mathbf{v} from the goal \mathbf{g} . While the robot follows the boundary of an obstacle, it stores the minimal distance $d_{\min}(\mathbf{g})$ to the goal, which it has observed so far along the explored part of the obstacle boundary, in order to determine a leave point. The transition phase ensures convergence of the robot to the goal.

Despite the algorithm's simplicity, TangentBug provides for provable convergence and completeness guarantees. An in-depth analysis of the algorithmic properties of the TangentBug algorithm can be found in Kamon et al. (1995, 1998). Under the same assumptions of Kamon et al. (1995, 1998), we restate the convergence of TangentBug in the following lemma.

Lemma 5.1. (Convergence of TangentBug Algorithm) *The TangentBug algorithm converges globally toward a reachable goal position inside a given planar domain for a sensor of any range in a finite path.*

Proof. Convergence directly follows from Theorem 1 and Theorem 2, proved in the work by Kamon et al. (1995) for a contact sensor. The proof in case of a non-zero range sensor follows the lines of the proofs for the contact sensor, and it can similarly be shown that the robot reaches the goal in a finite path if the goal's reachability is given (refer to Theorem 1 and Theorem 2, as formulated in the work by Kamon et al. (1998)).

5.3 Voronoi Coverage in Nonconvex Environments

In this section, a coverage solution is presented that extends Voronoi coverage to nonconvex environments with obstacles. The key idea is to combine Voronoi coverage with path planning. The path planning algorithm in combination with the Voronoi coverage algorithm let the robots converge to a constrained centroidal Voronoi configuration, and thereby solve a constrained optimization problem. Local path planning as well as global path planning algorithms can be used. We provide a detailed study of an implementation that uses a local path planner to compute the motion of the robots around obstacles and corners of a nonconvex environment.

5.3.1 Problem Formulation

A group of n robots must be deployed in the workspace $\mathcal{W} \subset \mathbb{R}^N$. The free space of the environment, $\Omega \subset \mathcal{W}$, is bounded and can either be convex, i.e., forming a convex domain without any obstacles or holes in it, or it can be

nonconvex, i.e., shaped by free-standing obstacles or holes and areas with nonconvex boundaries. Our focus is on nonconvex domains but a solution must also always be applicable to convex domains.

The original Voronoi coverage method formulates an optimization problem with a nonconvex aggregate objective function on a convex domain Ω , which is defined as a convex polytope (see Section 3.5.3). Here we restate the optimization problem from Equation (3.28) as the minimization with n set constraints,

$$\min_{\mathbf{P}} \mathcal{H}_V(\mathbf{P}), \text{ s.t. } \mathbf{p}_i \in \Omega, \forall i = \{1, \dots, n\}. \quad (5.1)$$

Due to the specific form of the objective function $\mathcal{H}_V(\mathbf{P})$ and the fact that the centroid of a convex polytope never falls outside the polytope, the function's critical points $\mathbf{p}_i^* = \mathbf{c}_{V_i}, \forall i = \{1, \dots, n\}$, are naturally contained in the feasible set Ω . Consequently, the set constraint is never active and the problem is an unconstrained optimization problem in fact. The objective function could now be changed, such that resulting gradients $\nabla \mathcal{H}_V(\mathbf{P})$ may point to the outside of the polytope as well. This would lead to a constrained optimization problem on the convex domain Ω . For instance, such a modified objective function is studied by Carpin (2012) for a method that combines the coverage functional with a second additive term which—opposed to coverage—tries to maximize covertness. If, instead of changing the objective function, nonconvex domains are considered for Ω , we obtain a constrained optimization problem with nonconvex objective function on a nonconvex domain. The problem has the form of Equation (5.1) and the set constraints now play a much more important role.

Let us first directly apply the gradient descent controller from Equation (3.33), as it is used for the original Voronoi coverage method with Euclidean distance in convex domains, to the constrained problem. Each robot r_i drives on a straight line to its goal position \mathbf{g}_i , which is the centroid \mathbf{c}_{V_i} of its Voronoi region V_i . We are confronted with two types of critical configurations: (1) the robot position temporarily leaves the domain Ω during motion, i.e., the path goes through an obstacle, and (2) the final goal position $\mathbf{g}_i = \mathbf{c}_{V_i}$ lies outside the domain in an obstacle and cannot be reached.

In order to resolve these situations, an obstacle avoidance behavior is required, which guarantees that the robots circumnavigate the obstacles safely. With regard to the constrained optimization problem, that means that the set constraints must be activated.

5.3.2 Gradient Projection Controller

The method of gradient descent is inherent in the CVT and leads to the gradient descent controller for Voronoi coverage. Hence, the logical choice is to introduce the *gradient projection method* from optimization (refer to Luenberger and Ye (2008) for details). The gradient projection method finds a feasible direction of motion by projecting the negative gradient $-\nabla\mathcal{H}_V(\mathbf{P})$ in each iteration onto the constraint surface, which is defined by the active constraints.

The n set constraints $\mathbf{p}_i \in \Omega$ in nN dimensions can be written explicitly as nM inequality constraints, which gives us

$$\min_{\mathbf{P}} \mathcal{H}_V(\mathbf{P}), \quad \text{s.t. } \mathbf{G}(\mathbf{P}) \leq \mathbf{0}. \quad (5.2)$$

If the domain Ω has arbitrary shape, it is bounded by a set of nM arbitrary nonlinear functions $\mathbf{G}(\mathbf{P}) = [G_j(\mathbf{P})]_{j=1}^{nM}$, and the inequality constraints are $\mathbf{G}(\mathbf{P}) \leq \mathbf{0}$. In polygonal environments, the domain Ω is represented as a nonconvex polytope and the inequality constraints are all linear. By setting $\mathbf{G}(\mathbf{P}) = \mathbf{A} \mathbf{P} - \mathbf{b}$, we get the inequality constraints $\mathbf{A} \mathbf{P} \leq \mathbf{b}$, with the $nM \times nN$ matrix \mathbf{A} , where each row defines a hyperplane and associated half-space, $a_{i,1}^j p_{i,1} + \dots + a_{i,N}^j p_{i,N} \leq b_i^j$, with $i \in \{1, \dots, n\}$, $j \in \{1, \dots, M\}$, and \mathbf{b} a vector of appropriate offsets. The active constraints are the subset of M_α inequality constraints for which strict equality holds, i.e., $\mathbf{G}_\alpha(\mathbf{P}) = \mathbf{0}$ or $\mathbf{A}_\alpha \mathbf{P} = \mathbf{b}_\alpha$, respectively. It defines a constraint surface $\mathcal{S}_\alpha \subset \partial\Omega$, on which to move in order to find a lower cost solution. This constraint surface is described at a regular point \mathbf{P} by the $(nN - M_\alpha)$ -dimensional tangent subspace, which can be represented in nN dimensions by the set $\mathbf{Q} = [\mathbf{q}_i]_{i=1}^{nN} \in \mathbb{R}^{nN}$, such that $\nabla \mathbf{G}_\alpha(\mathbf{P}) \mathbf{Q} = \mathbf{0}$.

In the following, we apply an *active set method* and, at all times, only consider the working set, i.e., the subset of constraints that are currently active. At each iteration during optimization the constraints are checked. Constraints which become active are included and constraints which have become inactive are excluded from the set of constraints. The working set is updated and the dimension of the constraint surface might be lowered or increased. By applying the gradient projection method, we seek a feasible direction vector \mathbf{z} that satisfies $\nabla\mathcal{H}_V(\mathbf{P}) \mathbf{z} < 0$, such that a movement along the direction of vector \mathbf{z} decreases the coverage cost $\mathcal{H}_V(\mathbf{P})$. The direction vector $\mathbf{z} \in \mathbf{Q}$ is obtained by projecting the negative gradient $-\nabla\mathcal{H}_V(\mathbf{P})$ onto the tangent subspace. Following the derivations by Luenberger and Ye (2008), one obtains the $nN \times nN$ projection matrices as

$$\mathbf{M}_{\text{proj}} = \mathbf{I} - \nabla \mathbf{G}_\alpha(\mathbf{P})^\top (\nabla \mathbf{G}_\alpha(\mathbf{P}) \nabla \mathbf{G}_\alpha(\mathbf{P})^\top)^{-1} \nabla \mathbf{G}_\alpha(\mathbf{P}) \quad (5.3)$$

for general nonlinear constraints, and

$$\mathbf{M}_{\text{proj}} = \mathbf{I} - \mathbf{A}_\alpha^T (\mathbf{A}_\alpha \mathbf{A}_\alpha^T)^{-1} \mathbf{A}_\alpha \quad (5.4)$$

for polygonal environments with linear constraints. The direction vector is then computed as

$$\mathbf{z} = -\mathbf{M}_{\text{proj}} \nabla \mathcal{H}_V(\mathbf{P})^T. \quad (5.5)$$

If $\mathbf{z} \neq \mathbf{0}$, a feasible direction is directly obtained for the case with linear constraints, and the robots r_i move along $\mathbf{z}_i = [z_{i,1}, \dots, z_{i,N}]$ according to the gradient descent controller of Equation (3.22), $\mathbf{u}_i = k \hat{\mathbf{z}}_i$. In the case with nonlinear constraints, the found point on the tangent subspace must first be projected back in perpendicular direction onto the constraint surface \mathcal{S}_α , such that $\mathbf{G}_\alpha(\mathbf{P}) = \mathbf{0}$, and the robots follow the constraint surface to reach the projected goal point.

After Luenberger and Ye (2008), if the projected gradient becomes zero and thus $\mathbf{z} = \mathbf{0}$, the *Karush-Kuhn-Tucker criterion* is satisfied, providing the necessary condition for a local minimum, $\nabla \mathcal{H}_V(\mathbf{P}) + \lambda^T \nabla \mathbf{G}_\alpha(\mathbf{P}) = \mathbf{0}$. The evaluation of the Lagrange multipliers $\lambda(\mathbf{P}) = -(\nabla \mathbf{G}_\alpha(\mathbf{P}) \nabla \mathbf{G}_\alpha(\mathbf{P})^T)^{-1} \nabla \mathbf{G}_\alpha(\mathbf{P}) \nabla \mathcal{H}_V(\mathbf{P})^T$ and $\lambda(\mathbf{P}) = -(\mathbf{A}_\alpha \mathbf{A}_\alpha^T)^{-1} \mathbf{A}_\alpha \nabla \mathcal{H}_V(\mathbf{P})^T$, respectively, reveals if a constrained local minimum is found. If $\lambda_j \geq 0, \forall j = \{1, \dots, M_\alpha\}$, the method has indeed arrived at a minimum and terminates. Otherwise, the active constraint which corresponds to the most negative multiplier λ_j is relaxed and removed from the working set. The method continues with the updated working set.

The gradient projection method converges like the original gradient descent method in linear time. However, some additional computation is required. In the case with nonlinear constraints, the negative gradient results from the projection to the tangent subspace. The projection matrix must be recomputed at every new robot configuration \mathbf{P} . The direction vector is then projected from the tangent subspace onto the constraint surface. A search along the curve on the constraint surface into the direction of the projected negative gradient must be conducted. For the polygonal environment, the projection onto the tangent subspace is sufficient. It is further possible to avoid the full recomputation of the projection matrix by updating the matrix from the previous one (Luenberger and Ye, 2008).

As recognized by Carpin (2012), the projection matrix \mathbf{M}_{proj} is a block diagonal matrix with n blocks $\mathbf{M}_{\text{proj},i}$ of dimension $N \times N$. The projection matrix can be decomposed and the computation of the gradient projection distributed among n robots. Each robot r_i computes its own feasible

direction vector

$$\mathbf{z}_i = -\mathbf{M}_{\text{proj}, i} \nabla_{\mathbf{p}_i} f(\mathbf{p}_i, V_i)^T. \quad (5.6)$$

This leads together with Equation (3.22) to a *gradient projection controller*, i.e., the gradient projection method, similar to the method of gradient descent, results in a distributed coordination strategy.

The extension of the Voronoi coverage method to nonconvex domains as outlined above follows closely the gradient projection method from optimization. For the application in robotics, however, we found a loose interpretation to be more meaningful. One difficulty is to model the boundary of a domain as function constraints $\mathbf{G}(\mathbf{P})$. Many constraints may be required, which would slow down the problem solving process. In the case of nonlinear constraints, a feasible point has to be searched and a robot needs to be guided along the constraint surface. This requires some sort of search or path planning method. Moreover, as now both the objective function and the constraints are nonconvex, there are additional local minima introduced. Even though the gradient projection method finds the constrained local minima, such minima, as a direct result of the added constraints, might be arbitrarily far away from the global minimum.

Based on these considerations, our coverage solution implements the basic idea of the gradient projection controller but introduces a path planner in order to complement the Voronoi coverage method. The path planner moves a robot along a constraint surface if constraints are active and moves it away once the constraints become inactive. The next section presents the idea in more detail, and describes a specific implementation based on a local path planning algorithm.

5.3.3 Combining Voronoi Coverage with Path Planning

Our solution for Voronoi coverage of a nonconvex environment builds on the Lloyd's algorithm (see Section 3.3.1, Algorithm 1) and a path planning algorithm. The coverage solution is composed of two layers of abstraction: on the upper layer (*level 1*), Lloyd's algorithm provides goal updates based on the successive computation of Voronoi regions and their centroids, which become the new goal positions, while, on the lower layer (*level 2*), the path planning algorithm generates the robot's path to the next goal position and is moving the robot toward the goal. This can be formulated as a continuing sequence of two loops, *loop 1* and *loop 2*, on level 1 and level 2, executed in a distributed fashion on each of the robots.

The modularity of the coverage solution basically allows for the use of any path planner. The difference lies in the provided capability and re-

quired knowledge of an individual planner. Local path planners are mainly concerned with navigating a robot to a close next waypoint and avoiding collisions on the way going there; they typically support sensor-based navigation, and only need a local map or no map at all. Global path planners are able to plan longer and more optimized paths but come with additional cost, such as the cost of maintaining and sharing a map of the environment.

In the remainder of this section, we focus on a simple local path planning algorithm, which provides us with the desired obstacle avoidance behavior. We use the TangentBug algorithm, as described in Algorithm 3, to extend the original Voronoi coverage method. As we will see shortly, TangentBug is in principle nothing other than an implementation of the gradient projection method. The TangentBug algorithm serves as path planner on level 2. TangentBug assumes that the environment is 2D and represented by polygons. Therefore, we look exclusively at polygonal planar environments in the following. Although TangentBug only needs local knowledge of the obstacles, note that global knowledge of the density function ρ is assumed for the execution of Lloyd’s algorithm throughout the presented approach.

A description of the proposed coverage solution is detailed in Algorithm 4 and Algorithm 5. The `UpdateNeighborhood` function updates, depending on the input argument (`SENS` or `COM`), the robot’s own position based on the sensed environment, or the positions of the Voronoi neighbors in the neighborhood \mathcal{N}_i via communication. The nonconvex coverage algorithm in Algorithm 4 then computes the Voronoi region and its centroid, and calls the path planning algorithm in Algorithm 5 as subroutine. Each algorithm implements one of the two loops of the coverage solution.

For the specific description of the algorithm we need to introduce some new terminology. We take inspiration from Pavone et al. (2008), and introduce the concept of *virtual robots* to navigation and path planning. We distinguish for each robot r_i between *real generators* at position $\mathbf{p}_i^{\text{real}}$ and *virtual generators* at position $\mathbf{p}_i^{\text{virt}}$ as well as *real goals* at position $\mathbf{g}_i^{\text{real}}$ and *virtual goals* at position $\mathbf{g}_i^{\text{virt}}$. $\mathbf{p}_i^{\text{real}}$ represents the actual robot position \mathbf{p}_i , whereas $\mathbf{p}_i^{\text{virt}}$ is the desired virtual robot position in disregard of the obstacles in the environment, as if we were dealing with a convex environment. $\mathbf{g}_i^{\text{virt}}$ is the ideal goal position, which corresponds to the centroid of the Voronoi region V_i^{virt} that has been computed from $\mathbf{p}_i^{\text{virt}}$ at the last update of loop 1. $\mathbf{g}_i^{\text{real}}$ finally designates the actual goal position, which corresponds to the projected position \mathbf{g}'_i , obtained from projecting goal position $\mathbf{g}_i^{\text{virt}}$ back onto the constraint surface, i.e., back to the closest position in the feasible domain Ω . A *constrained Voronoi region* is defined as the subset of all the Voronoi regions V_i^{virt} for which the condition, $V_i^{\text{virt}} \cap \partial\Omega \neq \emptyset \wedge \mathbf{g}_i^{\text{virt}} \notin \Omega$, applies. It

Algorithm 4 Nonconvex Coverage Algorithm

Require: Set of n robots r_i , each at initial position \mathbf{p}_i^0 in Ω . Each robot r_i is provided with modules for localization, communication, obstacle detection or environment modeling, and path planning, as specified in Chapter 3. Some prior knowledge of $\rho(\mathbf{q})$ over Ω (or \mathcal{W}) is available. Initialization of generators at time $t_i = 0$: $\mathbf{p}_i^{\text{real}} \leftarrow \mathbf{p}_i^0$, $\mathbf{p}_i^{\text{virt}} \leftarrow \mathbf{p}_i^0$.

- 1: **loop** {Loop 1}
- 2: $\mathbf{p}_i^{\text{real}} \leftarrow \text{UpdateNeighborhood}(\text{SENS})$
- 3: $\{\mathcal{N}_i, \{\mathbf{p}_j^{\text{virt}}\}_{j=1}^{|\mathcal{N}_i|}\} \leftarrow \text{UpdateNeighborhood}(\text{COM})$
- 4: $V_i^{\text{virt}} \leftarrow \text{ComputeVoronoiRegion}(\mathbf{p}_i^{\text{virt}}, \{\mathbf{p}_j^{\text{virt}}\}_{j=1}^{|\mathcal{N}_i|})$
- 5: $\mathbf{c}_{V_i^{\text{virt}}} \leftarrow \text{UpdateGoal}(V_i^{\text{virt}})$
 \Rightarrow update *virtual goal* position: $\mathbf{g}_i^{\text{virt}} \leftarrow \mathbf{c}_{V_i^{\text{virt}}}$
- 6: $\mathbf{p}_i^{\text{virt}} \leftarrow \text{PlanPath}(\mathbf{p}_i^{\text{real}}, \mathbf{g}_i^{\text{virt}}, V_i^{\text{virt}}, \text{algorithm}^1)$
- 7: **end loop**
- 8: $V_i \leftarrow \text{ComputeVoronoiRegion}(\mathbf{p}_i^{\text{real}}, \{\mathbf{p}_j^{\text{real}}\}_{j=1}^{|\mathcal{N}_i|})$

¹ **algorithm** is a convergent standard navigation algorithm with local or global path planning capability. In our case, the TangentBug algorithm is used.

is part of the constrained minimization problem, and is used as condition to determine if the inequality constraints from Equation (5.2), imposed by the boundary of Ω , are active in a region V_i^{virt} .

The proposed nonconvex coverage algorithm executes Lloyd's algorithm using the virtual generators, i.e., the Voronoi regions and their centroids are computed based on the virtual generators, and the virtual goal positions $\mathbf{g}_i^{\text{virt}}$ are updated. The virtual goals may be contained inside obstacles and the virtual generators are able to freely pass through obstacles and occlusions. The robots attempt to reach the virtual goals by moving toward the real goal positions $\mathbf{g}_i^{\text{real}}$. Throughout execution of loop 2, the virtual generators are moving toward the virtual goals ($\mathbf{p}_i^{\text{virt}} \rightarrow \mathbf{g}_i^{\text{virt}}$) in a simulated virtual environment representation where obstacles and nonconvex segments of the boundary do not exist. In parallel, the robots approach the real goal positions in the real environment taking obstacles into account, i.e., the real generators are moving toward the real goals ($\mathbf{p}_i^{\text{real}} \rightarrow \mathbf{g}_i^{\text{real}}$).

Each robot computes the next virtual and real goal positions upon arrival at the current real goal position. The neighbors of each robot pretend to be on track for an ideal convex case and communicate their simulated virtual

Algorithm 5 Path Planning Algorithm

Require: Robot r_i provided with sensor readings (local planner) or environment models (global planner), which enable the obstacle avoidance behavior. The virtual goal $\mathbf{g}_i^{\text{virt}}$ is required as goal position for the path planner. Initialization of variable \mathbf{v} : $\mathbf{v} \leftarrow \mathbf{g}_i^{\text{virt}}$.

- 1: **loop** {Loop 2}
 - 2: **if** V_i^{virt} is a constrained Voronoi region **then**
 - 3: Project $\mathbf{g}_i^{\text{virt}}$ to point \mathbf{g}'_i onto $\partial\Omega$ and set $\mathbf{v} \leftarrow \mathbf{g}'_i$ // goal projection
 - 4: **end if**
 \Rightarrow update *real goal* position: $\mathbf{g}_i^{\text{real}} \leftarrow \mathbf{v}$
 - 5: Execute next motion step toward real goal $\mathbf{g}_i^{\text{real}}$ by applying obstacle avoidance to drive to next position \mathbf{p}_i
 \Rightarrow update *real generator* position: $\mathbf{p}_i^{\text{real}} \leftarrow \mathbf{p}_i$
 - 6: Simulate next motion step toward virtual goal $\mathbf{g}_i^{\text{virt}}$
 \Rightarrow update *virtual generator* position $\mathbf{p}_i^{\text{virt}}$
 - 7: **end loop**
 - 8: **return** virtual generator position $\mathbf{p}_i^{\text{virt}}$
-

generator positions to the robot. That leads to a situation where the robots update their own Voronoi region, centroid and thus their next goal based on the virtual positions $\mathbf{p}_i^{\text{virt}}$ of their neighbors, while each of the robots is trying to reach its objective and get as close as possible to the specified ideal goal position $\mathbf{g}_i^{\text{virt}}$.

The virtual generators and goals allow for the implementation of Lloyd’s algorithm in the presence of nonconvexity, and maintain convergence of the Voronoi coverage method. If the robots’ real positions $\mathbf{p}_i^{\text{real}}$ were used for the computation of the Voronoi tessellation in turn, or if the points were continuously projected onto the boundary during ongoing execution of the navigation algorithm ($\mathbf{p}_i^{\text{virt}} = \mathbf{p}_i^{\text{real}}$ and $\mathbf{g}_i^{\text{virt}} = \mathbf{g}_i^{\text{real}}$), Lloyd’s algorithm could be massively perturbed depending on the shape of an obstacle and cause unfavorable behavior (e.g., overly long paths) or undetermined robot configurations (e.g., oscillations). Therefore, we design the algorithm in a way such that the real and virtual positions remain loosely coupled until the end.

Once the robots have converged to a final configuration of a local minimum, a last Voronoi tessellation is computed before the nonconvex coverage algorithm terminates. This last step is required because there might be several active constraints remaining, which is a consequence of the goal

projection procedure. In this case, all the robots whose Voronoi region is a constrained Voronoi region are not located at the positions of their Voronoi regions' centroids. A final computation of the Voronoi tessellation improves the overall partition and guarantees that each region of dominance assigned to a robot is at least a Voronoi region with respect to the robot positions $\mathbf{p}_i = \mathbf{p}_i^{\text{real}}$. Whenever the position of robot r_i in the final configuration lies in the free space away from an obstacle, the robot finally succeeds in reaching its specified goal position, and $\mathbf{p}_i^{\text{real}} = \mathbf{g}_i^{\text{real}} = \mathbf{g}_i^{\text{virt}} = \mathbf{p}_i^{\text{virt}}$ holds.

If, during the coverage procedure, a virtual goal position remains inside Ω , i.e., $\mathbf{g}_i^{\text{virt}} = \mathbf{g}_i^{\text{real}}$, and there is no obstacle in-between $\mathbf{p}_i^{\text{real}}$ and $\mathbf{g}_i^{\text{real}}$, the virtual and real generators coincide, $\mathbf{p}_i^{\text{real}} = \mathbf{p}_i^{\text{virt}}$, no projection is needed, i.e., $\mathbf{M}_{\text{proj}, i} = \mathbf{I}_2$, and robot r_i follows the unmodified gradient descent direction, which corresponds to $\mathbf{z}_i = -\nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i)^T$. In the case of an entirely convex environment, the virtual and real positions simply reduce to single real positions at all times, and the algorithm results in exactly the same behavior as for Voronoi coverage of convex environments, i.e., the constraints are not active.

Regarding the gradient projection method introduced earlier, $\mathbf{g}_i^{\text{real}} = \mathbf{g}'_i$ relates to the new improved position found from the projection of the negative gradient $-\nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i)$ onto the constraint surface \mathcal{S}_α . In contrast to the gradient projection method, now \mathbf{g}'_i is no longer limited to the current local working set and, more generally, can be located anywhere along the boundary of the set constraint, $\partial\Omega$. Therefore, the projected position $\mathbf{g}'_i \in \partial\Omega \subset \Omega$ is the position that is globally closest to the unconstrained position, $\mathbf{g}_i^{\text{virt}}$. This leads to the *goal projection* procedure

$$\mathbf{g}'_i = \operatorname{argmin}_{\mathbf{q} \in \Omega} \|\mathbf{q} - \mathbf{g}_i^{\text{virt}}\|_2. \quad (5.7)$$

Under application of the goal projection, the centroidal Voronoi tessellation becomes a CCVT, and the centroids are the constrained mass centers, as defined in Section 3.3.1. The robots are said to converge to a *constrained centroidal Voronoi configuration*.

The goal projection procedure emerges from Bug algorithms like Tangent-Bug in a natural way (see Section 5.2). The motion-toward-goal behavior is a form of gradient descent. If there are no obstacles on a robot's way, it just follows the gradient descent direction. The boundary-following behavior on the other hand is a form of exploration of obstacle boundaries, which realizes a search for a new improved position along the constraining boundary. Both behaviors together implement locally a gradient projection method, and additionally guarantee the robot's global convergence to a goal position. In particular, consider the case when the final goal is contained in an obstacle;

TangentBug comes with a *reachability test*, where reachability is determined during the boundary-following behavior by just circling around the obstacle. If the exploration of the obstacle boundary is completed after one full circle without having found a leave point, the goal will be unreachable. In this case, according to step 3 in Algorithm 5, the goal position must be projected onto the obstacle boundary in an optimal way. In order to implement this goal projection procedure, TangentBug can be extended, such that the robots check boundary positions for optimality during the boundary-following behavior. The optimal position along the boundary is continuously updated and stored in memory by updating Equation (5.7). Finally, projecting the virtual goal position $\mathbf{g}_i^{\text{virt}}$ to the obstacle boundary $\partial\Omega$ in an optimal way, for the robot simply means to drive directly to the recorded position \mathbf{g}'_i .

5.3.4 Properties of the Nonconvex Coverage Algorithm

Let us start the analysis of the coverage solution, presented in Algorithm 4 and Algorithm 5, by restating our assumptions. The robots are assumed to be point robots that move in free space $\Omega \subset \mathbb{R}^{nN}$. Ω is bounded, and both Ω and the set of obstacles \mathcal{O} are polygonal. Furthermore, both the perimeter of the obstacles and the number of obstacles are finite. The density function ρ is defined over Ω and is a priori known. Finally, we operate under setup 1 with regard to Section 3.5.3, i.e., the function to measure distance is the Euclidean distance, $d(\mathbf{q}, \mathbf{p}_i) = \|\mathbf{p}_i - \mathbf{q}\|_2$, $\mathbf{q}, \mathbf{p}_i \in \Omega$.

Convergence

We prove convergence of the coverage solution for the implementation based on the Lloyd's algorithm and the TangentBug algorithm, and for the case of a planar environment, i.e., the environment is represented by free space $\Omega \subset \mathbb{R}^2$ and workspace $\mathcal{W} \subset \mathbb{R}^2$. Similar proofs can be given for the coverage solution for other dimensions $N \neq 2$, and cases where variations on the Lloyd's algorithm or a local path planner other than TangentBug¹ are used.

Proposition 5.2. (Convergence of Nonconvex Coverage Algorithm) *For a nonconvex domain $\Omega \subset \mathbb{R}^2$, Algorithm 4 and Algorithm 5, based on the Lloyd's algorithm and the TangentBug algorithm, cause the robots to converge to a constrained centroidal Voronoi configuration.*

¹If the coverage solution uses other Bug-like algorithms on level 2, such as Bug1 or Bug2 (LaValle, 2006), similar plans result and the proofs only deviate slightly. The TangentBug algorithm is designed for \mathbb{R}^2 ; however, a direct extension to \mathbb{R}^3 exists (Kamon et al., 1999). Other more abstract planners work even in N dimensions.

Proof. We give a proof by contradiction. Suppose that the robots do not converge to a constrained centroidal Voronoi configuration. That must be a result of: (1) the TangentBug algorithm does not converge, or (2) the Lloyd’s algorithm does not converge. (1) By Lemma 5.1, if TangentBug does not converge, some goal point is not reachable. But that contradicts the projection properties of the coverage solution in Algorithm 4 and Algorithm 5—namely, that a projection of positions to Ω always exists. (2) By Lemma 3.2 and Lemma 3.3, respectively, if Lloyd’s algorithm does not converge, an iteration takes infinite time. But that implies (1), which, as we have already shown, leads to a contradiction—namely, that TangentBug always converges to a fixed and reachable goal position. \square

Optimality

When a goal position lies outside the domain, $\mathbf{g}_i^{\text{virt}} \notin \Omega$, as soon as the corresponding virtual generator also leaves the environment, $\mathbf{p}_i^{\text{virt}} \notin \Omega$, the real position of the robot must be constrained to the domain’s boundary $\partial\Omega$. Let us now see how the points can be projected to the boundary in the following.

Given the coverage solution in Algorithm 4 and Algorithm 5, and that n robots converge to the fixed points $\mathbf{p}_i^* = \operatorname{argmin}_{\mathbf{p}_i \in \Omega} \|\mathbf{p}_i - \mathbf{g}_i^{\text{virt}}\|_2$, $\forall i \in \{1, \dots, n\}$, in the domain Ω , which are projections of the optimal goal positions $\mathbf{g}_i^{\text{virt}} \notin \Omega$, with $\mathbf{g}_i^{\text{virt}} = \mathbf{c}_{V_i^{\text{virt}}}$, in the Euclidean sense. Define the final configuration vector $\mathbf{P}^* = [\mathbf{p}_i^*]_{i=1}^n \in \Omega^n$. We show in the proposition below that \mathbf{P}^* minimizes the high dimensional optimization problem $\min_{\mathbf{P} \in \Omega^n} \|\mathbf{P} - \mathbf{G}\|$, where $\mathbf{G} = [\mathbf{g}_i^{\text{virt}}]_{i=1}^n$. Furthermore, we show that this implies that \mathbf{P}^* locally minimizes a constrained optimization problem closely related to Equation (5.1).

Proposition 5.3. (Optimality of Nonconvex Coverage Algorithm) *The final configuration of the robots has the following properties:*

- (1) *The position \mathbf{P}^* is closest to \mathbf{G} in \mathbb{R}^{nN} in the Euclidean sense, given the projection of $\mathbf{g}_i^{\text{virt}}$ to its closest constrained point \mathbf{p}_i^* in \mathbb{R}^N , $\forall i \in \{1, \dots, n\}$.*
- (2) *The final step of the coverage solution in Algorithm 4 and Algorithm 5, which is the goal projection, solves the constrained optimization problem of minimizing the coverage cost $\mathcal{H}_{\mathcal{V}}(\mathbf{P})$ for the resulting final Voronoi partition $\mathcal{V} = \{V_i^{\text{virt}}\}_{i=1}^n$ with $\mathbf{g}_i^{\text{virt}}$ as generators.*

Proof. First, we prove (1) by contradiction. From the projection of the goal positions results that $\|\mathbf{p}_i - \mathbf{g}_i^{\text{virt}}\|_2$ is minimized at \mathbf{p}_i^* , $\forall i \in \{1, \dots, n\}$. Suppose that $\|\mathbf{P}^* - \mathbf{G}\|_2$ is not minimized. Then there exists a $\|\tilde{\mathbf{P}} - \mathbf{G}\|_2$ such that $\|\tilde{\mathbf{P}} - \mathbf{G}\|_2 < \|\mathbf{P}^* - \mathbf{G}\|_2$, that is $(\|\tilde{\mathbf{p}}_1 - \mathbf{g}_1^{\text{virt}}\|_2^2 + \dots + \|\tilde{\mathbf{p}}_n - \mathbf{g}_n^{\text{virt}}\|_2^2)^{1/2} < (\|\mathbf{p}_1^* - \mathbf{g}_1^{\text{virt}}\|_2^2 + \dots + \|\mathbf{p}_n^* - \mathbf{g}_n^{\text{virt}}\|_2^2)^{1/2}$. Substituting all $\|\tilde{\mathbf{p}}_i - \mathbf{g}_i^{\text{virt}}\|_2$ but one by $\|\mathbf{p}_i^* - \mathbf{g}_i^{\text{virt}}\|_2$ leads to $(\|\mathbf{p}_1^* - \mathbf{g}_1^{\text{virt}}\|_2^2 + \dots + \|\tilde{\mathbf{p}}_i - \mathbf{g}_i^{\text{virt}}\|_2^2 + \dots + \|\mathbf{p}_n^* - \mathbf{g}_n^{\text{virt}}\|_2^2)^{1/2} < (\|\mathbf{p}_1^* - \mathbf{g}_1^{\text{virt}}\|_2^2 + \dots + \|\mathbf{p}_i^* - \mathbf{g}_i^{\text{virt}}\|_2^2 + \dots + \|\mathbf{p}_n^* - \mathbf{g}_n^{\text{virt}}\|_2^2)^{1/2}$. From that it follows that $\|\tilde{\mathbf{p}}_i - \mathbf{g}_i^{\text{virt}}\|_2^2 < \|\mathbf{p}_i^* - \mathbf{g}_i^{\text{virt}}\|_2^2$, $\forall i \in \{1, \dots, n\}$, which is a contradiction.

Now we prove (2). We can rewrite the coverage cost using the parallel axis theorem, as given by Equation (3.31), $\mathcal{H}_V(\mathbf{P}) = \sum_{i=1}^n J_{V_i^{\text{virt}}, \mathbf{c}_{V_i^{\text{virt}}}} + \sum_{i=1}^n M_{V_i^{\text{virt}}} \|\mathbf{p}_i - \mathbf{c}_{V_i^{\text{virt}}}\|_2^2$. The first term on the right side of the equation is constant for a fixed area V_i^{virt} and the mass $M_{V_i^{\text{virt}}}$ is also constant. Since $\mathbf{c}_{V_i^{\text{virt}}} = \mathbf{g}_i^{\text{virt}}$, $\text{argmin}_{\mathbf{P}} \mathcal{H}_V(\mathbf{P}) = \text{argmin}_{\mathbf{P}} \sum_{i=1}^n M_{V_i^{\text{virt}}} \|\mathbf{p}_i - \mathbf{c}_{V_i^{\text{virt}}}\|_2^2 = \text{argmin}_{\mathbf{P}} \sum_{i=1}^n \|\mathbf{p}_i - \mathbf{c}_{V_i^{\text{virt}}}\|_2^2$, which is implied by the projection. \square

5.4 Voronoi Coverage in Unknown Environments

We limit ourselves in this section to a brief description of the transformation of nonconvex environments to star-shaped domains, which can be seen as a reformulation of the idea presented by Pimenta et al. (2008) as a robot-centric transformation. We provide a geometric as well as a mathematical interpretation of the transformation. The transformation provides a general way of dealing with nonconvex environments, and has successfully been applied in our work to extend the DisCoverage exploration algorithm of Haumann et al. (2010) to nonconvex environments, as presented by Haumann et al. (2011).

5.4.1 DisCoverage and Star-Shaped Domains

The reader refers to Haumann et al. (2010, 2011) for a presentation of the DisCoverage algorithm for multi-robot exploration and coverage. In addition, Haumann et al. (2011) includes simulations and experiments with e-puck robots that demonstrate the exploration of nonconvex environments under the transformation to star-shaped domains.

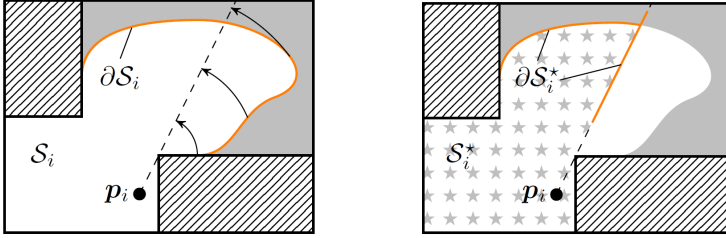


Figure 5.1: Transformation to star-shaped domain. Left: Nonconvex polygonal environment with explored domain \mathcal{S}_i . Right: Transformation $\mathcal{T}_{\mathbf{p}_i}$, generating the star-shaped domain \mathcal{S}_i^* .

The idea is to find a transformation with the property that all points $\mathbf{q} \in \mathcal{S}$ are visible from the robot position $\mathbf{p}_i \in \mathcal{S}$. This property holds for star-shaped domains. The transformation that maps any connected domain \mathcal{S} to a star-shaped domain \mathcal{S}^* is defined for robot r_i as follows.

Definition 5.1. Let $\mathcal{S}_i \subset \mathbb{R}^2$ and $\mathcal{S}_i^* \subset \mathbb{R}^2$. Let $d(\mathbf{p}_i, \mathbf{q})$ be the geodesic distance, or shortest path distance, from the robot position \mathbf{p}_i to \mathbf{q} , $\mathbf{p}_i, \mathbf{q} \in \mathcal{S}_i$. Let $\hat{\mathbf{w}}$ be the unit vector pointing into the direction of the first path segment of the geodesic path from \mathbf{p}_i to \mathbf{q} . Then, the transformation is given by $\mathcal{T}_{\mathbf{p}_i}: \mathcal{S}_i \rightarrow \mathcal{S}_i^*$, $\mathbf{q} \mapsto \mathbf{q}^* = \mathcal{T}_{\mathbf{p}_i}(\mathbf{q}) = \mathbf{p}_i + d(\mathbf{p}_i, \mathbf{q}) \hat{\mathbf{w}}$.

The map $\mathcal{T}_{\mathbf{p}_i}$ transforms any connected set to a star-shaped domain with respect to \mathbf{p}_i . The map $\mathcal{T}_{\mathbf{p}_i}$ can be interpreted as a *straightening* or *unwrapping* of the geodesic path. Note that $\mathcal{T}_{\mathbf{p}_i}$ is not bijective, i.e., it may map arbitrarily many elements from \mathcal{S}_i to only one element in \mathcal{S}_i^* . A geometric interpretation is provided by the illustration in Figure 5.1.

A significant property of $\mathcal{T}_{\mathbf{p}_i}$ is that elements $\mathbf{q} \in \mathcal{S}_i$ which are visible from \mathbf{p}_i remain unchanged, i.e., if it holds for the closed segment that $\overline{\mathbf{p}_i\mathbf{q}} \subset \mathcal{S}_i$, it immediately follows that $\mathcal{T}_{\mathbf{p}_i}(\mathbf{q}) = \mathbf{q}$. This is due to the fact that the geodesic path reduces to the shortest path in the Euclidean sense for convex environments.

Furthermore, the star-shaped environment \mathcal{S}_i^* , shown in Figure 5.1 on the right, is not necessarily a subset of the original environment \mathcal{S}_i , shown in Figure 5.1 on the left, since there exists no upper bound for the length of the geodesic distance.

As we will see in Chapter 6, this transformation also applies in discrete space and on graph representations of the environment. On a graph, the vector $\hat{\mathbf{w}}_{il}$, which points into the direction of the first path segment of the

geodesic path, connects a vertex v_i with one of its neighboring vertices $v_{n_{il}}$ in the one ring neighborhood $\mathcal{N}^{\text{ring}}$. This can be considered as the discrete equivalent of the continuous case above with $\hat{\mathbf{w}}$ pointing along the first segment toward a reflex vertex at a corner of a polygonal domain. A graph can even be imagined—as a thought experiment—to be realized by a heavily constrained continuous polygonal environment, where obstacles reduce the free space to a combination of lines (equivalents of edges) and intersection points (equivalents of vertices) only. Hence, following an analogous consideration as in this section, discrete domains on a graph can be transformed.

5.5 Results

For the simulations and experiments, we implemented the nonconvex coverage algorithm of Algorithm 4 and Algorithm 5 in Matlab. Algorithm 5 is implemented as the TangentBug algorithm after Kamon et al. (1998). The range sensors of the robots, which are a requirement of the TangentBug algorithm, are simulated, or emulated for the robots respectively, in software. In the following simulations and experiments, we use range sensors with infinite sensor range (the range is visualized by green rays), i.e., the sensors cover the whole visible area. For the sake of clarity in presentation, a uniform density function ρ is used. The density function is assumed to be known.

5.5.1 Evaluation of the Nonconvex Coverage Algorithm

Simulation Results

We present simulations in 2D and 3D environments. The robots are modeled as holonomic point robots.

2D Environment with Nonconvex Boundary. Figure 5.2 shows the deployment of five robots in a U-shaped environment. The robots cover the environment and converge to a final configuration, which is a centroidal Voronoi configuration in this case. The TangentBug algorithm implements the obstacle avoidance behavior and guides the robots around the corners of the obstacles.

Figure 5.3 presents the total coverage cost \mathcal{H}_V for the robot configuration. The cost is once computed for the Voronoi tessellation constructed from the real generators and once for the Voronoi tessellation constructed from the virtual generators. It is interesting to see how the cost for the real generators approaches the cost for the virtual generators over time. The virtual

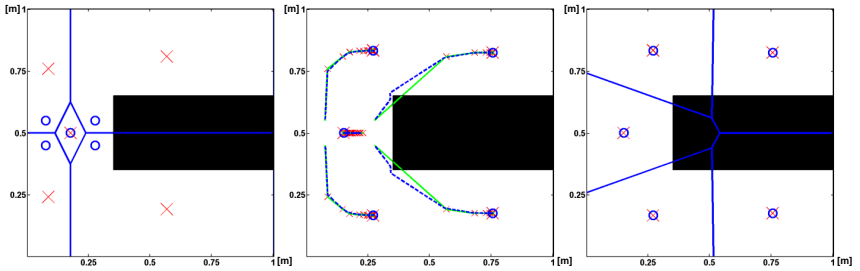


Figure 5.2: Voronoi coverage in a U-shaped environment. From left to right: five robots (blue circles) start from an initial position and move along the shown trajectories (dashed blue lines) to the next real goal position (red crosses) until they converge to a final configuration. Two of the trajectories of the virtual generators (green solid lines) intersect the obstacle’s corners, which indicates that an obstacle avoidance behavior is required for successful coverage of the environment.

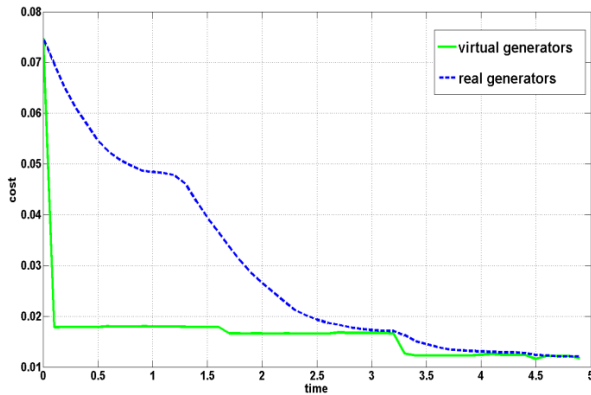


Figure 5.3: Cost for covering the U-shaped environment. The coverage cost \mathcal{H}_V for the configurations of the virtual and real generators in the U-shaped environment from Figure 5.2 is shown over time.

generators are updated by the classical Lloyd’s algorithm, i.e., the virtual generators are directly set to the positions of the newly computed centroids. As soon as a robot reaches its real goal position, the Voronoi region, the virtual goal and the virtual generator are updated, and the cost for the virtual generators falls off.

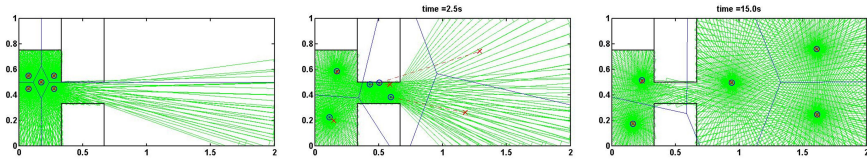


Figure 5.4: Voronoi coverage in an environment with narrow passage. From left to right: Five robots start in the small room on the left, three of the five robots move through the passage and deploy in the right larger room.

In a second experiment, five robots cover an environment that is divided by a narrow passage (see Figure 5.4). All the robots start from one side. During the coverage procedure, three of the five robots transition through the passage and deploy in the right side of the environment. This simulation demonstrates that environments with narrow passages can be covered successfully by the proposed coverage solution. However, we note that the robots’ initial positions influence the coverage result. The nonconvex coverage algorithm, similar to the original Voronoi coverage method, is not guaranteed to generate equitable partitions (Pavone et al., 2008).

2D Environment with Free-Standing Obstacles. Five robots cover an area with two free-standing obstacles in Figure 5.5. The virtual generator and the virtual goal of one of the robots are contained in an obstacle. The robot explores the obstacle and updates the goal projection continuously. After the robot has completed one full cycle, it drives to the position on the obstacle boundary that is closest to the virtual goal position. At convergence, a constrained centroidal Voronoi configuration is reached. A last Voronoi tessellation is computed in order to improve the final partition and reduce the resulting cost this way once more.

Another example of an environment with free-standing obstacle is shown in Figure 5.6. Here, the obstacle is itself nonconvex. Different from the previous simulation, where the virtual goal remained inside the obstacle at the end, the virtual generators and virtual goals of two of the five robots are contained inside the obstacle temporarily during deployment but have returned to the free space when the final robot configuration is reached.

Nonconvex 3D Environment. The proposed nonconvex coverage algorithm does, similar to the original Voronoi coverage method, also extend to three and even higher dimensions. This is based on the fact that Lloyd’s

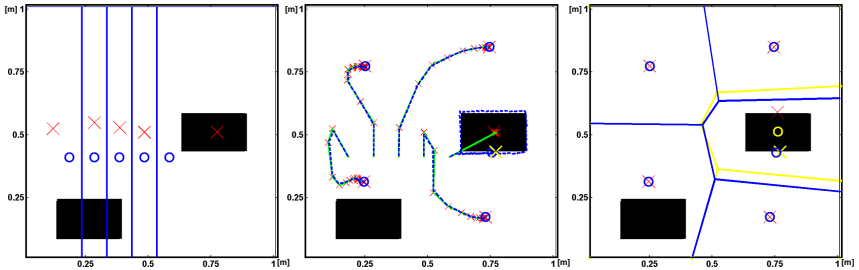


Figure 5.5: Voronoi coverage in an environment with two free-standing obstacles. The virtual generator and goal of one robot lie inside an obstacle. The robot tries to reach them and starts circling around the obstacle. Finally, the goal is projected to the closest point on the obstacle's boundary (yellow cross). The centroidal Voronoi tessellation (solid yellow lines) results from the virtual generators at the virtual goal positions, whereas the Voronoi diagram in blue represents the last improved partition computed by the robots in a final step.

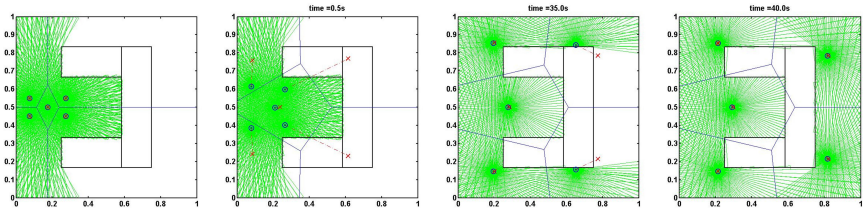


Figure 5.6: Voronoi coverage in an environment with one nonconvex free-standing obstacle. From left to right: Five robots start to cover the environment. The virtual goals of two robots are contained inside the obstacle. Therefore the robots circle the obstacle to explore its boundary and find the best projection of their virtual goals. By updating the Voronoi partition, the two virtual goals return to the free space and the robots converge to a centroidal Voronoi configuration.

algorithm as well as standard path planning algorithms can be applied in three and higher dimensions.

Even though a full 3D version of the TangentBug algorithm can be implemented, we once more use the 2D version, and limit ourselves to a demonstration of the coverage solution in a 3D environment which is free of non-convexities in one dimension. Figure 5.7 shows five flying robots that cover a 3D U-shaped environment.

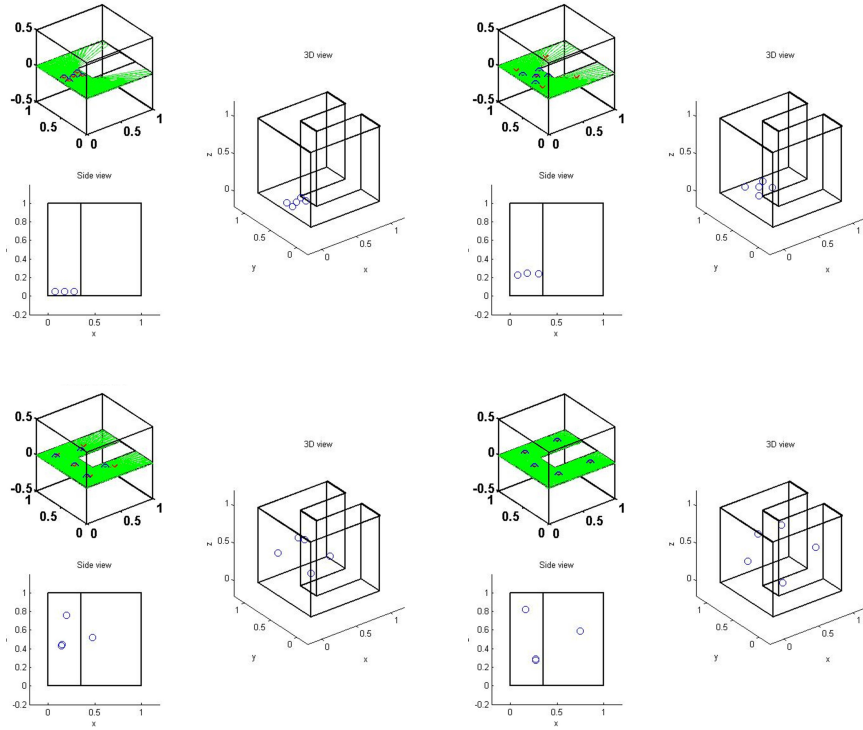


Figure 5.7: Voronoi coverage in a 3D nonconvex environment. Five flying robots take off and deploy in 3D space, which is convex in the vertical direction.

Experimental Results

The coverage solution for nonconvex environments is demonstrated in the following experiments with five e-puck robots. We used the planar test setup, as presented in Section 2.1.2, and localized the robots with an overhead camera. The obstacles and environment boundaries were overlaid on the ground plane as virtual obstacles and boundaries and the occurrence of collisions between robots and obstacles were checked through computer vision. In contrast to simulated robots, the e-puck robots have a finite physical size and are non-holonomic. We checked the direction of a robot movement for collisions and dilated the detected obstacle and environment boundaries by the size of a robot's radius. The experiments show the robustness of the coverage solution in asynchronous operation and against communication delays.

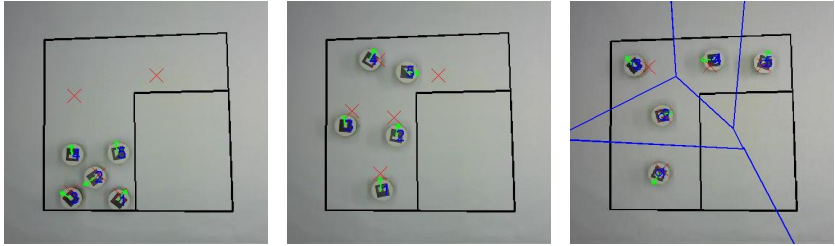


Figure 5.8: Voronoi coverage in an L-shaped environment. From left to right: five e-pucks start from the bottom and gradually spread over the free-space by avoiding the corner. A centroidal Voronoi configuration results at convergence.

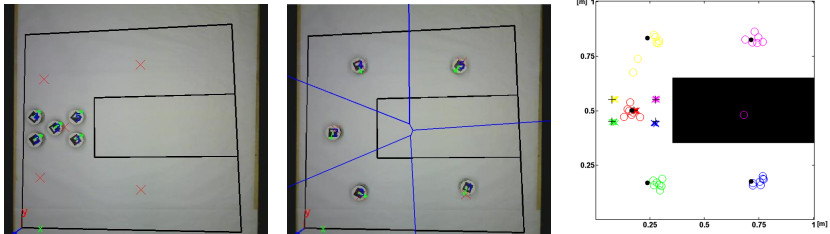


Figure 5.9: Voronoi coverage in a U-shaped environment. From left to right: five e-pucks avoid the obstacle and cover the nonconvex environment. The plot on the right shows the initial and final configurations over the seven test runs. Ideal simulated positions in black (cross: initial position, dot: final position) and real experimental positions in color (cross: initial positions, circle: final positions). The magenta circle inside the obstacle shows a failed experimental run where the tracker lost the marker of one robot.

Nonconvex 2D Environment. Figure 5.8 and Figure 5.9 show experimental runs in an L-shaped and U-shaped environment with five e-puck robots. The robots succeed in both cases to cover the nonconvex environment. For the U-shaped environment, the initial robot positions and the final configuration are given in Figure 5.9; we ran seven experimental trials with the robots for the given initial positions in the U-shaped environment. The experimental results match with the simulations. While hardware noise and tracking errors only cause small deviations, a main difference in the trajectories comes from adjustments in the algorithm to account for the non-zero size of the real robots by a safety margin along the boundary. The average

position error over the robots and the experimental runs is 5.42 cm. The duration of one experimental run is 4.56 min in average. Though the convergence of the robots toward the final configuration was limited by the update rate of the tracking system rather than by the robot platform or the control strategy itself.

5.6 Summary

In this chapter, we present a coverage solution to provide Voronoi coverage in nonconvex environments. Our nonconvex coverage algorithm combines the Lloyd's algorithm with a path planning algorithm: Lloyd's algorithm updates the Voronoi partition and the current goal position, while the path planner computes a feasible path around obstacles and corners to the goal. This addresses both objectives of coverage and obstacle avoidance at the same time.

We prove convergence and optimality of the proposed coverage solution by applying the concept of virtual robots to multi-robot path planning. Through virtual generators and goal positions, a way for decoupling the robot positions and the Voronoi partitions is introduced.

The nonconvex coverage algorithm is evaluated in simulations and physical experiments with a group of e-puck robots. We use the TangentBug algorithm, a derivative of the family of Bug algorithms, as path planner in our implementation and show that TangentBug realizes a gradient projection method. The projection procedure constrains goal positions to the free space of the environment; the projected positions are solutions to the constrained optimization problem of finding a (constrained) centroidal Voronoi configuration in the nonconvex environment. Constrained optimization more generally allows for covering environments of mixed dimensions, where robots move partly unconstrained and partly constrained to a lower dimensional space. An example would be a robot that flies freely, and then lands and climbs a surface.

An alternative method to handle nonconvexity in Voronoi coverage is the use of a geodesic distance function. We use such a distance for the transformation to a star-shaped domain, which unwraps the shortest path to the points in a Voronoi region hidden behind corners and obstacles. In particular, based on this transformation, we designed an exploration algorithm which is closely linked to Voronoi coverage.

The approaches of this chapter are focused on the continuous space. The next chapter will study discrete representations. Global graph-based path planners, such as the A* search, will be used in place of the TangentBug algorithm in combination with Voronoi coverage.

Chapter 6

Multi-Robot Coverage on Curved Surfaces

We extend the Voronoi coverage method to curved surfaces in this chapter. Our coverage solutions use a *discrete representation* of the environment. The presented coverage algorithms deploy multiple robots into discrete partitions over the curved surfaces by operating on a graph. We describe an implementation that embeds the graph into 3D space and uses a triangle mesh, which represents a standard technique to model surfaces and 3D objects in computer graphics. As a discrete representation of the environment is used for computation, it is a reasonable question to ask what we could gain from an explicit formulation and direct study of the discrete problem. In Chapter 5, we have seen the relation between graph-based representations and heavily constrained continuous workspaces. A graph, such as a triangle mesh, can but does not necessarily need to include the obstacles and nonconvexities of a workspace in the representation. If obstacles are not modeled and left out, they are handled implicitly, since a robot is always driven along a sequence of vertices, faces or edges contained within the graph. Another positive side effect of the graph-based representation is that it offers a direct link to computational geometry and computer graphics, which provide a great variety of useful concepts and algorithms for geometric problems, such as robot surface coverage.

We present two distributed adaptive coverage algorithms for Voronoi coverage on curved surfaces, as they are typically found in inspection applications for tanks and tubes (see also Chapter 1 and Section 2.1 in Chapter 2). The algorithms are executed directly on the graph and represent discretized versions of the continuous formulations in Section 3.5.3 of Chapter 3 and Chapter 5.

Findings from the previous chapter, such as the shortest path distance and the related transformation to star-shaped domains, or the concept of virtual generators and goal positions, will come up again. The first coverage algorithm computes shortest path distances and propagates a discrete wavefront on the graph to obtain a centroidal graph Voronoi tessellation. The second coverage algorithm approximates distances on the surface through the Euclidean distance in the ambient space and locally exchanges mesh elements between adjacent Voronoi regions to create a final centroidal Voronoi configuration. In the generalized adaptive versions of the algorithms, a metric tensor field is additionally computed locally on the surface; it is used to shape the Voronoi regions in position, size, orientation and aspect ratio according to the present anisotropy. Both algorithms are compared and evaluated in simulations on different mesh models and in experiments with five e-puck robots on a curved surface.

Some of the concepts and results in this chapter have been presented at the International Conference on Intelligent Robots and Systems (Breitenmoser et al., 2010a) and at the International Symposium on Distributed Autonomous Robotic Systems (Breitenmoser et al., 2012). We start with Section 6.1 on related work. Information about the graph-based representation is provided by Section 6.2. Section 6.3 presents the problem formulation and the two proposed coverage solutions. We then discuss the extensions of the algorithms to adaptive and hybrid coverage control in Section 6.4. Results from simulations and experiments are described in Section 6.5, and final conclusions are drawn in Section 6.6.

6.1 Related Work

In mobile robotics, the robots' environment is commonly represented by a planar grid or elevation map with the strong underlying assumption of a 2D or 2.5D world (Hebert et al., 1989). Although this is a valid assumption for many of the practical applications, there are environments, including outdoor rough terrain, urban areas or industrial structures, which require true 3D representations. A polygonal mesh, as it is used in computer graphics for modeling of 3D objects, is a good candidate; the works by Rusu et al. (2009) and Gingras et al. (2010) are examples for the use of triangle meshes for environment representation in robotics.

A mesh can be seen as a graph. Graphs have widely been used in robot path planning on grid maps or for topological representations, e.g., in roadmaps and communication or pose graphs, but have not that often been applied in the context of triangle meshes.

The recent works by Bhattacharya et al. (2012), Durham et al. (2012), and Yun and Rus (2012), all use graph-based representations for multi-robot control, and are closely related to Voronoi coverage and our own work. The multi-robot coverage method of Bhattacharya et al. (2012) extends the work by Pimenta et al. (2008) to anisotropic metrics and non-Euclidean metric spaces. Such spaces, for example, result from charting a curved parametric surface. The CVT constructed in the coordinate chart accounts for the surface curvature, and mapping the tessellation back to the original surface finally results in a proper centroidal Voronoi configuration on the curved surface. This approach and our first coverage algorithm are similar in the way that both rely on a shortest path distance, as used by Pimenta et al. (2008), and propagate a wavefront on the graph. However, our focus is on curved surfaces of arbitrary geometry embedded in 3D space; our representations are triangle meshes and our algorithm runs directly on the mesh.

The coverage methods of Durham et al. (2012) and Yun and Rus (2012) are related to our second coverage algorithm. They mainly differ from our solution in the way how two adjacent Voronoi regions are updated. Besides, they primarily address the coverage of planar environments. Durham et al. (2012) implements a discrete Voronoi coverage method that works with unreliable pairwise communication. A pairwise partitioning rule updates two adjacent Voronoi regions by trying to achieve an *optimal pairwise partition* at each update step using exhaustive search. Yun and Rus (2012) applies a vertex substitution algorithm for the update. This considers information of the neighbors as well as the neighbors' neighbors of a robot, i.e., a two-hop communication is used, which achieves locally optimal configurations.

In computational geometry and computer graphics, CVTs are often applied for mesh generation and remeshing. The work by Du et al. (2002) generates a CCVT on curved surfaces by projecting the mass centers of the Voronoi regions onto the surface in each iteration step. The approach assumes knowledge of a parametrization of the surface and is thus limited in its application to simple standard surface geometries, for which such a parametrization is known. The method of Peyré and Cohen (2004) segments meshes based on the computation of CVTs and shortest geodesics; here, the shortest path distances are computed by fast marching methods. The method of Valette et al. (2008) approximates the CVT by using the Euclidean distance metric; the computed CVT is then applied to remeshing of 3D mesh models. Peyré and Cohen (2004) and Valette et al. (2008) both present centralized methods—nevertheless, we have taken some inspiration from these works with regard to decentralized multi-robot coverage. Du and Wang (2005b) represents a somewhat complementary approach compared to

the previous approach of Du et al. (2002). Instead of projecting the mass centers from the ambient space onto the surface, an ACVT is computed in a coordinate chart and mapped to the parametric surface, much like in the work by Bhattacharya et al. (2012).

Besides, the concept of anisotropy has been used in the method of Gusraldi et al. (2009) to model anisotropic sensors for Voronoi coverage. However, the anisotropy is formulated with respect to the robot positions and not with respect to points in the environment. Cortés et al. (2004) suggests to use the density function ρ from the CVT formulation as isotropic weight. The density influences the centroidal Voronoi configuration and allows for formation control and robot guidance. Finally, Pavone et al. (2009) creates fat- and skinny-shaped equitable partitions by using power diagrams, with applications to vehicle routing and optimal workload sharing.

6.2 Preliminaries

The surface is approximated by a triangle mesh M , which enables a graph-based representation of the surface. There are different approaches for generating the triangle mesh (see Section 7.2.2 for examples), and several methods for creating a graph and its embedding from the triangle mesh. We use the following two methods to create the graph in our work. In the first method, the mesh itself directly forms the embedded graph. The graph $G_{\text{mesh}} = \{V, E\}$ consists of vertices V , which are the vertices or corners of the triangles of the mesh, and edges E , which are the edges of the triangles. The second method creates the dual graph $G_{\text{mesh}}^* = \{V^*, E^*\}$ of G_{mesh} from the triangle mesh M . The vertices V^* are the centroids of the triangles of the mesh and the edges E^* connect two such centroids whenever the two corresponding triangles share a common edge. Figure 6.1 provides an illustration.

For some of the derivations below, the area associated to a single vertex of the graphs G_{mesh} and G_{mesh}^* is required. These areas are obtained for graph G_{mesh} (or G_{mesh}^*) as the areas of the faces of the dual graph G_{mesh}^* (or G_{mesh}). Under the first method, the area $A(v)$ of G_{mesh} is computed as the area of the corresponding face in graph G_{mesh}^* enclosing v . In the case of a planar mesh, this face can be viewed as the Voronoi region around v obtained from the Voronoi diagram with generator set V , i.e., the vertices of G_{mesh} , which are also the mesh vertices, act as the generators. In case of graph G_{mesh}^* , the area $A(v^*)$ is simply given by the area of the triangle in which v^* is located.

In the following, if not otherwise stated, we will not distinguish between graphs G_{mesh} and G_{mesh}^* , and refer to either of them by G_M . Note that, as a practical triangle mesh M is always finite, G_M is a finite graph. Having

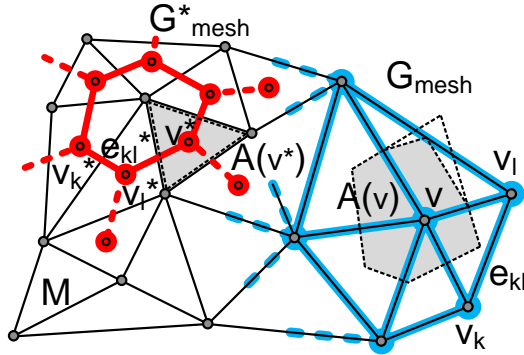


Figure 6.1: Graph-based representation. The graph G_M is created from the triangle mesh M , either as graph G_{mesh} or its dual graph G_{mesh}^* .

n robots r_i , in particular, it holds that $M_i \subset M$, $G_{M_i} \subset G_M$ and $G_i \subset G$. M_i denotes a robot's mesh map and G_{M_i} is the graph-based representation associated to M_i . If the mesh maps of two robot neighbors overlap, the meshes are merged, $M_i \leftarrow M_i \cup M_j$ and $M_j \leftarrow M_i \cup M_j$. G_i is the connected subgraph whose vertices form the robot's Voronoi region V_{G_i} . The graph G is partitioned according to a DCVT, $\mathcal{G}(V_G) = \{V_{G_i}\}_{i=1}^n$. G is a subgraph of G_M , $G \subset G_M$, and the Voronoi tessellation spreads over a subset of the overall known mesh M .

6.3 Voronoi Coverage on Curved Surfaces

Covering a curved surface embedded in 3D space is no straightforward extension of the planar case to higher dimensions. In this section, we present two different approaches on how to realize Voronoi coverage on curved surfaces, which will result in two coverage algorithms. Curved surfaces are of practical importance regarding the inspection task. Moreover, they are examples of curved spaces and, as such, they represent on their own an interesting object of research.

6.3.1 Problem Formulation

Given the workspace $\mathcal{W} \subset \mathbb{R}^3$ and a connected orientable curved surface $\mathcal{S} \subset \mathcal{W}$, which is a 2D Riemannian manifold with the Riemannian metric induced by the Euclidean scalar product, a group of n robots with positions

$\mathbf{p}_i \in \mathcal{S}$ is to be deployed over \mathcal{S} to cover the surface. The deployment must be in accordance with Voronoi coverage, i.e., the robots are distributed over a CVT of \mathcal{S} , which partitions the surface into n Voronoi regions $V_i \subset \mathcal{S}$, each containing one of the robots. That is the group of robots deploys and converges to a centroidal Voronoi configuration on the curved surface.

With respect to Equation (3.25), we consider the performance function $h(d(\mathbf{q}, \mathbf{p}_i)) = d_{\mathbf{q}, \mathbf{p}_i}^p$, with $\mathbf{q} \in \mathcal{S}$, and define the distance in \mathcal{S} by

$$d_{\mathbf{q}, \mathbf{p}_i} = \inf_{\substack{\gamma: [0, 1] \rightarrow \mathcal{S} \\ \gamma(0) = \mathbf{q}, \gamma(1) = \mathbf{p}_i}} \int_0^1 \|\mathbf{B}_{\gamma(t)} \dot{\gamma}(t)\|_p dt. \quad (6.1)$$

This is the usual geodesic distance of the *Finsler manifold*, which is given by \mathcal{S} together with the Finsler function $F: T\mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$, such that at position $\mathbf{x} \in \mathcal{S}$, $F_{\mathbf{x}}(\mathbf{z}) = \|\mathbf{B}_{\mathbf{x}} \mathbf{z}\|_p$. $\mathbf{B}_{\mathbf{x}}$ is called the *Finsler tensor field* in the following. $\mathbf{B}_{\mathbf{x}}$ is a symmetric positive definite matrix smoothly depending on \mathbf{x} , which describes the local anisotropy. Such anisotropy might reflect the distribution of curvature of the surface \mathcal{S} along different directions, or the varying importance of covering a certain direction on \mathcal{S} .

To determine the distance $d_G(v, v_{\mathbf{p}_i})$ for a given graph G_{M_i} , the integral in Equation (6.1) is discretized and approximated by the total length of a sequence of segments. The length is computed from the sum of distances,

$$d_{\mathbf{v}, \mathbf{v}_{\mathbf{p}_i}} = \sum_{l=1}^m d_{\mathbf{v}_{l-1}, \mathbf{v}_l} \simeq \sum_{l=1}^m \left\| \tilde{\mathbf{B}}_{\mathbf{v}_{l-1}} (\mathbf{v}_l - \mathbf{v}_{l-1}) \right\|_p = d_G(v, v_{\mathbf{p}_i}), \quad (6.2)$$

along the sequence of m segments $\mathcal{S}_{v_0, v_m} = \{\overline{\mathbf{v}_0 \mathbf{v}_1}, \overline{\mathbf{v}_1 \mathbf{v}_2}, \dots, \overline{\mathbf{v}_{m-1} \mathbf{v}_m}\}$, with $\mathbf{v}_0 = \mathbf{v}$ and $\mathbf{v}_m = \mathbf{v}_{\mathbf{p}_i}$. The distance of each single segment is evaluated by applying the directional distance from Section 3.3.1 for arbitrary p -norms.

Although \mathcal{S} is a 2D manifold, it is not sufficient in the discrete version to define $\mathbf{B}_{\mathbf{v}}$ on the 2D tangent spaces of \mathcal{S} because the distance vectors $(\mathbf{v}_l - \mathbf{v}_{l-1})$ are not restricted to the tangent spaces $T_{\mathbf{v}_{l-1}}\mathcal{S}$ at \mathbf{v}_{l-1} . So, however we choose the matrix $\mathbf{B}_{\mathbf{v}}$ on the tangent spaces, it must be extended to a 3×3 matrix $\tilde{\mathbf{B}}_{\mathbf{v}}$. This matrix can be constructed from the desired matrix $\mathbf{B}_{\mathbf{v}}$ given in a basis of $T_{\mathbf{v}}\mathcal{S}$, e.g., such that, in the local basis extended by the normal vector of the oriented surface, it looks like $\tilde{\mathbf{B}}_{\mathbf{v}} = \begin{pmatrix} \mathbf{B}_{\mathbf{v}} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}$.

The coverage cost for the group of n robots, computed over the triangle mesh M represented by graph G_M , is now derived. In line with Equa-

tion (3.25) and Equation (6.1) we can write

$$\mathcal{H}_{\mathcal{V}}(\mathbf{P}) = \sum_{i=1}^n \sum_{v \in V_{G_i}} \int_{A(v)} d_{\mathbf{q}, \mathbf{P}_i}^p \rho(\mathbf{q}) dF(\mathbf{q}) . \quad (6.3)$$

Under the approximation that $\tilde{\mathbf{B}}_v$ and the density function ρ are constant over a given vertex area $A(v)$, i.e., $M_{A(v)} = \int_{A(v)} \rho(\mathbf{q}) dF(\mathbf{q}) = A(v) \rho_G(v)$, and in addition, constraining the robot positions \mathbf{P} to \mathbf{P}_G on the graph, the coverage cost can be rewritten as

$$\mathcal{H}_G(\mathbf{P}_G) = \sum_{i=1}^n \sum_{v \in V_{G_i}} d_G(v, v_{\mathbf{P}_i})^p M_{A(v)} , \quad (6.4)$$

where $d_G(v, v_{\mathbf{P}_i})$ is the shortest path distance on the graph, as defined in Equation (6.2).

In order to minimize the coverage cost \mathcal{H}_G and thereby deploy the robots on the curved surface, we once more start from the Lloyd's algorithm (see Section 3.3.1, Algorithm 1). Lloyd's algorithm consists of the two steps of computing the CVT and updating the generator positions. Each of the two steps poses a subproblem that needs to be solved to arrive at a coverage solution. Both of our two approaches will eventually lead to coverage solutions which solve the two steps in the Lloyd's algorithm. In both of the approaches we rely on the triangle mesh M and the graph-based representations G_M , and work with variants of DCVTs on the graph. The *surface coverage algorithm* in Algorithm 6 provides the overall framework of the coverage solutions. The surface coverage algorithm is divided into a *coordination* and a *moving* action. The coverage algorithms of the two approaches under consideration are called by the `Coordinate` function on line 3. In each iteration, after the coordination action has computed the Voronoi region and updated the goal direction or goal position, respectively, a robot moves toward the updated goal under the moving action. The `UpdateNeighborhood` function accesses information from the environment in a robot's neighborhood \mathcal{N} and provides an update on the robot positions and mesh maps. At the end of each iteration, the `UpdateState` function is called to update the state and action that are to be performed in the next iteration.

Algorithm 6 Surface Coverage Algorithm

Require: Set of n robots r_i , each at initial position \mathbf{p}_i^0 on \mathcal{S} in the continuous domain. The corresponding discrete position $\mathbf{v}_{\mathbf{p}_i}^0$ on the initial graph $G_{M_i}^0$, with $G_{M_i}^0$ the graph associated to the initial mesh M_i^0 , and initial regions $G_i^0 \subset G_{M_i}^0$. Each robot r_i is provided with modules for localization, communication, environment modeling and path planning, as specified in Section 3.1.

```

1: while state == DEPLOY do
2:   if action == COORDINATE then
3:      $\mathbf{v}_{\mathbf{g}_i} \leftarrow \text{Coordinate}(\text{approach})$  // (1) “wavefront propagation”, or
                                                // (2) “vertex exchange”
4:   end if
5:   if action == MOVE then
6:      $\{\mathbf{p}_i, M_i\} \leftarrow \text{UpdateNeighborhood}(\text{SENS})$ 
7:      $T_i \leftarrow \text{PlanPath}(\mathbf{v}_{\mathbf{p}_i}, \mathbf{v}_{\mathbf{g}_i}, G_{M_i}, M_i)$ 
8:      $\mathbf{p}_i \leftarrow \text{MoveToGoal}(\mathbf{p}_i, T_i)$ 
9:      $\text{UpdateState}()$ 
10:  end if
11: end while

```

6.3.2 Surface Coverage with Shortest Path Distance

Our first approach is the more straightforward of the two. We use the triangle mesh as a graph, and let the robots construct a DCVT directly on the graph in a distributed fashion. This leads to a coverage solution that is intrinsic in its very nature and only depends on the local surface geometry and graph connectivity.

In our two approaches to the problem of covering a curved surface, we use different distance computation. We vary the manner the sequence of vertices v_l is chosen in defining $d_{\mathbf{v}, \mathbf{v}_{\mathbf{p}_i}}$ after Equation (6.2). For this first approach, $d_{\mathbf{v}, \mathbf{v}_{\mathbf{p}_i}}$ is computed for arbitrary p -norms as the discrete geodesic distance or shortest path distance on the graph, $d_G(v, v_{\mathbf{p}_i})$. The distance segments correspond to the edges along a shortest path that connects v and $v_{\mathbf{p}_i}$ on the graph G_{M_i} . This corresponds to setup 2 in Section 3.5.3, and represents a natural way to measure distance over a curved surface (see also Section 3.2.1).

In the following, the coverage solution under the first approach is presented. Derivations and implementation details for solving both of the two steps of computing the CVT and updating the generator positions in the Lloyd’s algorithm are described.

Gradient Descent Controller for Curved Spaces

We first look at the update of the robot positions \mathbf{P} , and postpone the computation of the DCVT on the graph to the next section. In principle, we want to use the gradient descent controller from Equation (3.21) to move each robot toward the centroid of its Voronoi region, as we have done before. Now dealing with curved spaces, one needs to be cautious, since the gradient of the aggregate objective function for a robot r_i , $\nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i)$, does not represent the vector that points directly into the direction of the minimum of the coverage cost $h(\mathbf{p}_i, V_i)$ in general (see also Equation (3.24) and Equation (3.29)). We assume in the following the existence of only one shortest geodesic. The direct direction to the cost minimum corresponds to the direction of the shortest geodesic, which is in fact the shortest path connecting the robot position with the centroid of the Voronoi region of the current Voronoi tessellation. Equivalence between the gradient direction and the direction along the shortest geodesic only holds for the special case of Euclidean space.

This can better be seen from the following considerations. We look for a vector \mathbf{y} in the direction in which the coverage cost $h(\mathbf{p}_i, V_i)$ for robot r_i increases fastest, and then move into the opposite direction along vector \mathbf{z} . Each robot has its own *Riemannian robot motion metric* $\mathbf{R}_{\mathbf{p}_i}$, which is again extended to $\tilde{\mathbf{R}}_{\mathbf{p}_i}$, a measure which captures the cost for a robot of moving in a certain direction. We now write

$$\max_{\mathbf{y}} \nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i) \mathbf{y}, \quad \text{s.t.} \quad \sqrt{\mathbf{y}^T \tilde{\mathbf{R}}_{\mathbf{p}_i} \mathbf{y}} = 1, \quad (6.5)$$

which is solved under the requirement of $\mathbf{y} \neq \mathbf{0}$ by

$$\nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i) \frac{\partial}{\partial \mathbf{y}} \left(\frac{\mathbf{y}}{\sqrt{\mathbf{y}^T \tilde{\mathbf{R}}_{\mathbf{p}_i} \mathbf{y}}} \right) = \mathbf{0}^T. \quad (6.6)$$

We then get as result for the final vector \mathbf{z}

$$\mathbf{z} = -\mathbf{y}, \quad \text{with} \quad \mathbf{y} = c_i \tilde{\mathbf{R}}_{\mathbf{p}_i}^{-1} \nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i)^T, \quad (6.7)$$

where c_i is a constant. There are two solutions, a maximum and a minimum value, with either coefficient $c_i = c_{i,1}$ or $c_i = c_{i,2}$. As $\tilde{\mathbf{R}}_{\mathbf{p}_i}$ is positive definite, the maximum is obtained for the coefficient which satisfies $c_i > 0$. By selecting this c_i , we are able to design a gradient descent controller in the very same manner as before according to Equation (3.22), $\mathbf{u}_i = k \hat{\mathbf{z}}$, with positive gain k . The included correction term $\tilde{\mathbf{R}}_{\mathbf{p}_i}^{-1}$ in Equation (6.7) accounts for the influence of the local anisotropy on the robot r_i caused by the curved space.

Above findings, however, do not alter the basic requirement of computing the gradient of the coverage cost, $\nabla_{\mathbf{p}_i} \hat{h}(\mathbf{p}_i, V_i)$, which is finally needed to provide an update of the robot position. In the following, we describe two possible methods for the computation of such a gradient.

Explicit Gradient Computation. A first method is to compute the gradient $\nabla_{\mathbf{p}_i} \hat{h}(\mathbf{p}_i, V_i)$ at each robot position \mathbf{p}_i explicitly. In the discrete setting, robot positions are represented by graph vertices $v_{\mathbf{p}_i}$, and differences between coverage cost terms at neighboring vertices on the graph can be computed,

$$\Delta \hat{h}_{il} := \frac{\hat{h}(\mathbf{v}_{n_{il}}, V_{G_i}) - \hat{h}(\mathbf{v}_{\mathbf{p}_i}, V_{G_i})}{\|\mathbf{v}_{\mathbf{p}_i, n_{il}}\|_2}, \quad (6.8)$$

with $\mathbf{v}_{\mathbf{p}_i, n_{il}} = \mathbf{v}_{n_{il}} - \mathbf{v}_{\mathbf{p}_i}$. The term $\hat{h}(\mathbf{v}_{\mathbf{p}_i}, V_{G_i})$ denotes the coverage cost for a robot at position \mathbf{p}_i , indicated by vertex $v_{\mathbf{p}_i}$, and $\hat{h}(\mathbf{v}_{n_{il}}, V_{G_i})$ is the coverage cost for the unchanged DCVT on the graph, $\mathcal{G}(V_G)$, except that the robot position and the associated vertex $v_{\mathbf{p}_i}$ have moved to a position that is represented by the neighboring vertex $v_{n_{il}}$. Given m neighboring vertices $v_{n_{il}}$ of vertex $v_{\mathbf{p}_i}$, a possible solution is to approximate the gradient vector by least squares fitting, using the m vectors $\hat{\mathbf{v}}_{\mathbf{p}_i, n_{il}}$ as data points. The resulting gradient vector follows from the overdetermined system of equations, $\mathbf{W} \mathbf{h}_i = \mathbf{y}$, in usual manner, as the best fit $\mathbf{h}_i^* = \mathbf{W}^+ \mathbf{y}$. \mathbf{W}^+ is the pseudoinverse $(\mathbf{W}^T \mathbf{W})^{-1} \mathbf{W}^T$, where \mathbf{W} is the $m \times 3$ matrix with the m vectors $\hat{\mathbf{v}}_{\mathbf{p}_i, n_{il}}^T$ in its rows, and the m -dimensional vector \mathbf{y} is composed of the m coverage cost differences $\Delta \hat{h}_{il}$. This solution provides only a rough approximation, since the gradient computation is increasingly degraded with growing irregularity of the triangle mesh M . Irregularities arise when many edges in the mesh, or graph G_M respectively, point into the same direction, whereas other directions are left without any edges. Alternative solutions for gradient estimation are studied in literature. Most of the works assume regular grids; among others, a study on linear gradient estimation methods for unstructured meshes can be found in the work by Correa et al. (2011).

The resulting gradient vector \mathbf{h}_i^* is then corrected according to Equation (6.7) to obtain $\mathbf{z} = -c_i \tilde{\mathbf{R}}_{\mathbf{v}_{\mathbf{p}_i}}^{-1} \mathbf{h}_i^*$. As we are on a graph or mesh, respectively, the vector \mathbf{z} is finally restricted to the closest graph edge or mesh face.

Gradient Computation by Shortest Geodesics. Alternatively, the robot position can be updated from the tangent directions of the shortest geodesics from the current robot position \mathbf{p}_i to points \mathbf{q} in the robot's Voronoi region. From the gradient of the coverage cost in Equation (3.29), we

see that the partial derivative of the performance function $h(d(\mathbf{q}, \mathbf{p}_i))$ needs to be computed. With $h(d(\mathbf{q}, \mathbf{p}_i)) = d(\mathbf{q}, \mathbf{p}_i)^p$, we get $\nabla_{\mathbf{p}_i} h(d(\mathbf{q}, \mathbf{p}_i)) = pd(\mathbf{q}, \mathbf{p}_i)^{p-1} \nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$, and the only part missing to determine the gradient $\nabla_{\mathbf{p}_i} h(\mathbf{p}_i, V_i)$ is the partial derivative of the distance function, $\nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$.

Next, we derive $\nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$ for the general continuous case, and for arbitrary p -norms. We denote the negative gradient of the shortest path distance $d(\mathbf{q}, \mathbf{p}_i) = d_{\mathbf{q}, \mathbf{p}_i}$ with $\mathbf{d}_i := -\nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$ in the following. We now look at the shortest geodesic $\gamma^*(t)$ from $\gamma^*(0) = \mathbf{q}$ to $\gamma^*(1) = \mathbf{p}_i$. The negative tangent vector of the geodesic at point \mathbf{p}_i is oriented toward $\gamma^*(0)$ and is denoted as \mathbf{w} , i.e., $\mathbf{w} := -\dot{\gamma}^*(1)$. We observe that the absolute change $|\mathbf{d}_i|$ in the distance of the shortest geodesic along the tangent direction \mathbf{w} must be maximal; it results in a maximum decline. If the decline of the shortest geodesic distance was maximal in a different direction than the tangent direction \mathbf{w} , this would immediately lead to a new shortest geodesic from this direction, which is a direct consequence of the principle of optimality (LaValle, 2006). But this is in contradiction with the fact that the shortest geodesic leads along \mathbf{w} , and \mathbf{d}_i indeed results in a maximal decline along the tangent direction \mathbf{w} . Therefore, we can write in a similar way to Equation (6.5) above,

$$\max_{\mathbf{w}} \mathbf{d}_i \mathbf{w}, \text{ s.t. } \|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}\|_p = 1. \quad (6.9)$$

Requiring $\mathbf{w} \neq \mathbf{0}$, we solve the equation

$$\mathbf{d}_i \frac{\partial}{\partial \mathbf{w}} \left(\frac{\mathbf{w}}{\|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}\|_p} \right) = \mathbf{0}^T, \quad (6.10)$$

and finally find the direction of the negative gradient \mathbf{d}_i ,

$$\mathbf{d}_i = C \left(|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}|^{p-1} \cdot \text{sgn}(\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}) \right)^T \tilde{\mathbf{B}}_{\mathbf{p}_i}, \quad (6.11)$$

with a positive constant C . Note that the absolute value, the exponentiation and the “.”-operator for multiplication in $\left(|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}|^{p-1} \cdot \text{sgn}(\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}) \right)$ all are applied component-wise to the vectors. In order to get not only the direction but also the value of \mathbf{d}_i , we look at an infinitesimal change in the distance of Equation (6.1). From the first order Taylor approximation, we can see that it must hold

$$d_{\mathbf{q}, \mathbf{p}_i} + \mathbf{d}_i \mathbf{w} \simeq d_{\mathbf{q}, \mathbf{p}_i} + \|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}\|_p. \quad (6.12)$$

We finally arrive at the partial derivative of the distance function

$$\nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i) = - \frac{\left(|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}|^{p-1} \cdot \text{sgn}(\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}) \right)^T \tilde{\mathbf{B}}_{\mathbf{p}_i}}{\|\tilde{\mathbf{B}}_{\mathbf{p}_i} \mathbf{w}\|_p^{p-1}}. \quad (6.13)$$

For the common choice of $p = 2$, we get with $\tilde{\mathbf{K}}_{\mathbf{p}_i} := \tilde{\mathbf{B}}_{\mathbf{p}_i}^T \tilde{\mathbf{B}}_{\mathbf{p}_i}$,

$$\nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)^\top = - \frac{\tilde{\mathbf{K}}_{\mathbf{p}_i} \mathbf{w}}{\sqrt{\mathbf{w}^\top \tilde{\mathbf{K}}_{\mathbf{p}_i} \mathbf{w}}} . \quad (6.14)$$

In the discrete case on the graph, matrices $\tilde{\mathbf{B}}_{\mathbf{p}_i}$ and $\tilde{\mathbf{K}}_{\mathbf{p}_i}$ are given for the discrete positions at the graph vertices, and the tangent vectors \mathbf{w} are exchanged for the vectors \mathbf{w}_{il} along the first segments of the shortest path distances from $v_{\mathbf{p}_i}$ to vertices v on the graph, i.e., along the graph edges connecting the vertex of robot position $v_{\mathbf{p}_i}$ with the corresponding vertices v_{ni} of its one ring neighborhood $\mathcal{N}_i^{\text{ring}}$. This is clearly an approximation of the continuous case, since vectors \mathbf{w}_{il} —different from vectors \mathbf{w} at \mathbf{p}_i —are not anymore contained in the tangent space $T_{\mathbf{p}_i} \mathcal{S}$.

The vector \mathbf{z} in direction of the cost minimum can now be calculated for the continuous and discrete versions, and for arbitrary p -norms, by using Equation (6.7) and the relation in Equation (3.29) with $h(d(\mathbf{q}, \mathbf{p}_i)) = d(\mathbf{q}, \mathbf{p}_i)^p$,

$$\begin{aligned} \mathbf{z}_{\text{cont}} &= - c_i p \tilde{\mathbf{R}}_{\mathbf{p}_i}^{-1} \int_{V_i} d(\mathbf{q}, \mathbf{p}_i)^{p-1} \nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)^\top \rho(\mathbf{q}) dF(\mathbf{q}) , \\ \mathbf{z}_{\text{disc}} &= - c_i p \tilde{\mathbf{R}}_{\mathbf{v}_{\mathbf{p}_i}}^{-1} \sum_{v \in V_{G_i}} d(\mathbf{v}, \mathbf{v}_{\mathbf{p}_i})^{p-1} \nabla_{\mathbf{p}_i} d(\mathbf{v}, \mathbf{v}_{\mathbf{p}_i})^\top M_{A(v)} , \end{aligned} \quad (6.15)$$

where $\nabla_{\mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$ is obtained from Equation (6.13) and $d(\mathbf{q}, \mathbf{p}_i)$ is given by $d_{\mathbf{q}, \mathbf{p}_i}$ after Equation (6.1). In the discrete case, the tangent vectors \mathbf{w} are replaced by \mathbf{w}_{il} , the distance $d(\mathbf{v}, \mathbf{v}_{\mathbf{p}_i})$ is given by $d_G(v, v_{\mathbf{p}_i})$ after Equation (6.2) and the density defined over the vertex area $A(v)$ is again contracted to a single value per vertex. Furthermore, the resulting vector \mathbf{z}_{disc} of the discrete version needs to be projected onto the graph.

Even in the case $p = 2$, the two matrices $\tilde{\mathbf{R}}_{\mathbf{x}}$ and $\tilde{\mathbf{K}}_{\mathbf{x}}$ are not equal in general. Whereas $\tilde{\mathbf{R}}_{\mathbf{x}}$ is the robot motion metric, $\tilde{\mathbf{K}}_{\mathbf{x}}$ describes how distance along different directions is measured in the construction of the CVT. In the special case where the matrices are equal, $\tilde{\mathbf{R}}_{\mathbf{x}}$ and $\tilde{\mathbf{K}}_{\mathbf{x}}$ cancel each other out in the numerator of Equation (6.14). $\tilde{\mathbf{K}}_{\mathbf{x}}$ only remains in the denominator and implicitly within the distance function $d(\mathbf{q}, \mathbf{p}_i)$.

Let us assume $p = 2$, cancelation of $\tilde{\mathbf{R}}_{\mathbf{v}_{\mathbf{p}_i}}$ and $\tilde{\mathbf{K}}_{\mathbf{v}_{\mathbf{p}_i}}$, and uniform metric tensor fields everywhere, i.e., $\mathbf{K}_{\mathbf{v}} = \mathbf{I}_2$ and $\tilde{\mathbf{K}}_{\mathbf{v}} = \mathbf{I}_3$, respectively. We obtain

as the basic form of the discrete version of Equation (6.15),

$$\mathbf{z}_{\text{disc}} = 2 c_i \sum_{v \in V_{G_i}} d_G(v, v_{\mathbf{p}_i}) \hat{\mathbf{w}}_{il} M_{A(v)}, \quad (6.16)$$

which is consistent with the results by Cortés et al. (2004) for Voronoi coverage in 2D convex environments (refer also to Section 3.5.3), by Pimenta et al. (2008) for generalized Voronoi coverage in 2D nonconvex environments and by Bhattacharya et al. (2012) for higher dimensional non-Euclidean spaces.

We would like to elaborate on one simple approximative way of updating the robot positions in the following. Under the assumptions that the metric is isotropic and the triangle mesh M and graph G_M are fairly regular, the evaluation by summing over the direction vectors in Equation (6.15) or Equation (6.16) can be substituted for summing over weights of the vertices $v_{n_{il}}$ in the one ring neighborhood of $v_{\mathbf{p}_i}$ instead. The weight of each $v_{n_{il}}$ is computed by counting the number of times a shortest path leaves from $v_{\mathbf{p}_i}$ to a vertex v in the graph while passing over $v_{n_{il}}$, and accumulating the lengths computed for these paths for each $v_{n_{il}}$. The accumulated path lengths are the sums $\sum_{(v \in V_{G_i} \mid v_{n_{il}} \in \mathcal{S}_{v_{\mathbf{p}_i}, v})} D(v)$. $\mathcal{S}_{v_{\mathbf{p}_i}, v}$ is the sequence of vertices in the shortest path from $v_{\mathbf{p}_i}$ to v , and $D(v)$ denotes a robot's distance map, which holds the lengths of the shortest path from each v to the robot position $v_{\mathbf{p}_i}$, $d_G(v, v_{\mathbf{p}_i})$. The new goal position $\mathbf{v}_{\mathbf{g}_i}$ results as the position of the vertex $v_{n_{il}}^*$ in the one ring neighborhood with maximum accumulated path length. $\mathbf{v}_{\mathbf{g}_i}$ is the vertex in the one ring neighborhood the robot encounters first on the shortest path to the intrinsic center of mass of the current Voronoi region.

A permanent decrease in the local coverage cost $h(\mathbf{v}_{\mathbf{p}_i}, V_{G_i})$ through transitions from $\mathbf{v}_{\mathbf{p}_i}$ to $\mathbf{v}_{\mathbf{g}_i}$ is assured by requiring $h(\mathbf{v}_{\mathbf{g}_i}, V_{G_i}) < h(\mathbf{v}_{\mathbf{p}_i}, V_{G_i})$. The permanently decreasing coverage cost is important to guarantee the convergence of the coverage procedure. If several vertices $v_{n_{il}}$ in the one ring neighborhood with maximum accumulated path lengths result, possible solutions are to select one by random or to compare their coverage cost and to select the one that results in the highest decrease of cost. When a resulting $\mathbf{v}_{\mathbf{g}_i}$ does not lead to a decrease in the coverage cost, we conclude that the robot is currently located at the centroid of its discrete Voronoi region, and the robot is not moved in the current iteration.

In the case of a strictly regular grid, such as a planar rectangular or hexagonal grid map, the new goal vertex can alternatively be obtained by only counting the number of times a path passes a vertex $v_{n_{il}}^*$ when leaving from $v_{\mathbf{p}_i}$. In addition, the calculation can be done for each of the directions independently (e.g., for a rectangular 4-connected grid, in horizontal and

vertical directions). The new goal vertex is then found by comparing all the counts that resulted for each of the independent grid directions.

Before we continue with the next section, we want to put the two possible methods for gradient computation in relation to each other. The explicit gradient computation is more intuitive at first glance, however, it is computationally costly and good estimates of the gradient directions are not easy to compute for unstructured irregular meshes. In contrast, the gradient computation by shortest geodesics is appealing as most of the work for computing the shortest path distances has already been completed for constructing a DCVT on the graph (see the following section). Therefore, this gradient computation nearly comes for free. Furthermore, as just seen above, the gradient directions can even be computed intrinsically on the graph for given approximations, without the need for any additional geometric information from the graph embedding.

Coordination by Wavefront Propagation

We now describe the computation of the DCVT on the graph under the first approach. We show how the DCVT computation, together with the update of the generator positions from the previous section, composes the first coverage algorithm for Voronoi coverage on the surface. An overview of the first coverage algorithm is given in Algorithm 7, which is called by Algorithm 6. Each robot first communicates with its neighboring robots to update the neighborhood information; positions and mesh maps are exchanged.

The `PropagateWavefront` function computes discrete geodesic distances between vertices by propagating wavefronts over the graph. During runtime, each robot r_i keeps track of the vertices in the graph G_{M_i} by using the following data structures: a priority queue with a sorted candidate list L_{front} of the vertices at the propagating wavefront, a distance map D with the lengths of the shortest path from each vertex v to the robot position on the graph, an identity map I with the identity of the robot closest to v and a map from each vertex v to the vertex in the one ring neighborhood $\mathcal{N}_i^{\text{ring}}$ of the closest robot which lies on the shortest path assigned to vertex v . Hence, for each vertex v in a Voronoi region V_{G_i} , the shortest path distance from $v_{\mathbf{p}_i}$ to v , the first vertex v_{next} of $\mathcal{N}_i^{\text{ring}}$ on the shortest path passing from $v_{\mathbf{p}_i}$ to v , as well as the identity i of the assigned Voronoi region are stored.

Under the coordination action, a robot begins the execution of one iteration of the Lloyd's algorithm. The robot considers the positions $v_{\mathbf{p}_j}$ of the neighboring robots on the graph one after the other by initiating a wavefront propagation that proceeds until the wavefront reaches a neighboring robot (Figure 6.2, left). For the wavefront propagation, a label correcting method

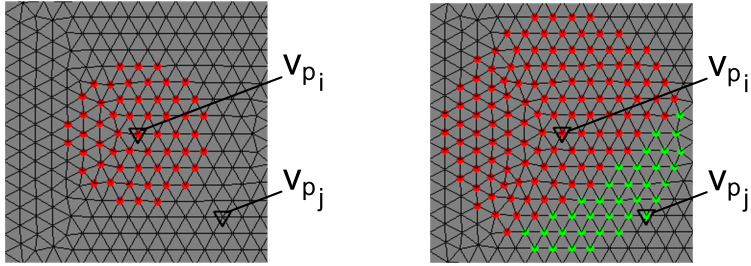


Figure 6.2: Wavefront propagation. Left: A wavefront propagation is initiated by robot r_i from vertex v_{p_i} on the mesh. Right: After a neighboring robot r_j has been reached, a back propagation is started from vertex v_{p_j} . The back propagation stops when the reverse wavefront becomes closer to robot r_i than to robot r_j again; by appropriate label correction, the Voronoi region V_{G_i} and its graph G_i of robot r_i are incrementally constructed.

Algorithm 7 Coordination by Wavefront Propagation

Require: Voronoi region G_i , graph G_{M_i} and mesh M_i .

- 1: $\{\mathcal{N}_i, \{v_{p_j}\}_{j=1}^{|\mathcal{N}_i|}, G_{M_i}, M_i\} \leftarrow \text{UpdateNeighborhood}(\text{COM})$
 - 2: $G_i \leftarrow \text{PropagateWavefront}(\mathcal{N}_i, \{v_{p_j}\}_{j=1}^{|\mathcal{N}_i|}, G_i, G_{M_i}, M_i)$
 - 3: $\mathbf{v}_{g_i} \leftarrow \text{UpdateGoalDirection}(G_i)$
 - 4: $\text{UpdateState}()$
-

based on the Dijkstra's algorithm is used (see Section 3.3.2). Thereby, D and I are initially generated. At the point of reaching a neighbor robot r_j , the propagation is paused and a *back propagation* starts on the visited vertices. The back propagation terminates when the vertices at the reverse wavefront are going to be closer to robot r_i than to its neighbor r_j again (Figure 6.2, right). This is verified by simply checking against $D(v)$. All the vertices which are closer to the neighbor robot are then reassigned to the neighbor robot, which corrects the labeling of the identity map I . After the back propagation ends, the wavefront propagation continues until the next neighbor of robot r_i is reached, where another back propagation starts. Or, the algorithm terminates as the priority queue L_{front} is empty, i.e., $D(v)$ has been calculated for all remaining vertices in the graph. Alternatively, termination is triggered through an abort criterion. Our abort criterion is based on the technique introduced by Cortés et al. (2004), where the ball around robot

r_i is iteratively enlarged up to the minimal size that enables to compute the Voronoi region completely. In the discrete setting, the abort criterion for robot r_i on the graph G_{M_i} is

$$\frac{D(v_L)}{2} \geq \max_{\substack{v \in G_{M_i}, \\ I(v) = I(v_{\mathbf{p}_i})}} d_G(v, v_{\mathbf{p}_i}), \quad \forall v_L \in L_{\text{front}}. \quad (6.17)$$

In the second step of the Lloyd's algorithm, the robot positions are updated by closely following the concepts of the previous section. Once the Voronoi region V_{G_i} has been constructed, the shortest path distances from the robot's position $v_{\mathbf{p}_i}$ to all the vertices v contained in the robot's Voronoi region are available. The coverage cost of Equation (6.3), or Equation (6.4) respectively, can then be evaluated by a summation of these distance values. The new goal vertex $\mathbf{v}_{\mathbf{g}_i}$ is selected among the vertices of the one ring neighborhood $\mathcal{N}_i^{\text{ring}}$ with the help of Equation (6.15) or Equation (6.16), respectively. The vector \mathbf{z} is projected onto the graph. The robot then switches to the moving action in Algorithm 6. A gradient descent controller after Section 3.5.2 moves the robot to $\mathbf{v}_{\mathbf{g}_i}$, and closer toward the centroid of its Voronoi region. The robot position on the graph and the Voronoi regions are updated in each iteration and a permanent decrease in the coverage cost \mathcal{H}_G results through the robot movements.

6.3.3 Surface Coverage with Approximative Euclidean Distance

The second approach presents a coverage solution that follows an approximative approach, which has relations to clustering. Neighboring robots exchange vertices of their Voronoi regions V_{G_i} among each other and deploy into an optimal centroidal Voronoi configuration on the curved surface. From the viewpoint of a multi-robot system, this implements a distributed coordination. With regard to Voronoi tessellations, a DCVT with Euclidean distance metric is constructed by this second approach. The path distance $d_{\mathbf{v}, \mathbf{v}_{\mathbf{p}_i}}$ from Equation (6.2) is approximated by the length of a single segment, which connects \mathbf{v} and $\mathbf{v}_{\mathbf{p}_i}$ along the direct shortcut through \mathbb{R}^3 , using the 2-norm. We arrive this way at the directional distance $d_{\mathbf{v}, \mathbf{v}_{\mathbf{p}_i}} \simeq \left\| \tilde{\mathbf{B}}_{\mathbf{v}}(\mathbf{v}_{\mathbf{p}_i} - \mathbf{v}) \right\|_2 = \sqrt{(\mathbf{v}_{\mathbf{p}_i} - \mathbf{v})^T \tilde{\mathbf{K}}_{\mathbf{v}}(\mathbf{v}_{\mathbf{p}_i} - \mathbf{v})}$, where $\tilde{\mathbf{K}}_{\mathbf{v}} := \tilde{\mathbf{B}}_{\mathbf{v}}^T \tilde{\mathbf{B}}_{\mathbf{v}}$, as introduced in Section 3.3.1, and similar to setup 1 under Section 3.5.3. This distance is a usual anisotropic quadratic distance in \mathbb{R}^3 .

Including the ambient space explicitly for the computation of the Voronoi tessellation has several implications. The computed centroids may be con-

tained in the ambient space and need to be restricted onto the surface \mathcal{S} and the graph G_M . From this perspective, the approach resembles the procedures of Chapter 5 for computing CCVTs, where the virtual generators and goal positions must be projected back onto the feasible set. As the ambient space is an integral part of the approach, the coverage solution inevitably depends on the graph embedding. In addition, the approach builds on several approximations; the approximations affect the solution's accuracy but have the potential for an overall speed up in computation when compared with the coverage solution of the first approach.

The starting point into the discussion of the second approach is a detailed explanation of the approximative DCVT formulation described in the next section. It is followed by the presentation of the second coverage algorithm, which again addresses the two aforementioned steps of CVT computation and updating the generator positions in the Lloyd's algorithm.

Approximative DCVT for Curved Spaces

We look at Equation (6.3) and want to simplify the expression with the overall objective of reducing the computational effort which is needed for recomputing the CVT at every iteration of Lloyd's algorithm. First, let us revisit the parallel axis theorem stated in Theorem 3.1 and formally given in Equation (3.30). For general non-Euclidean metrics, we can write under setup 1

$$\begin{aligned} J_{A(v), \mathbf{p}_i} &= \int_{A(v)} \|\tilde{\mathbf{B}}_{\mathbf{q}}(\mathbf{p}_i - \mathbf{q})\|_2^2 \rho(\mathbf{q}) dF(\mathbf{q}) \\ &= J_{A(v), \mathbf{c}(v)} + \int_{A(v)} \|\tilde{\mathbf{B}}_{\mathbf{q}}(\mathbf{p}_i - \mathbf{c}(v))\|_2^2 \rho(\mathbf{q}) dF(\mathbf{q}) \quad (6.18a) \end{aligned}$$

$$\simeq J_{A(v), \mathbf{c}(v)} + M_{A(v)} \|\tilde{\mathbf{B}}_{\mathbf{c}(v)}(\mathbf{p}_i - \mathbf{c}(v))\|_2^2, \quad (6.18b)$$

where the total mass over the area $A(v)$ and the area's mass center are given by

$$M_{A(v)} = \int_{A(v)} \rho(\mathbf{q}) dF(\mathbf{q}) \quad , \quad \text{and} \quad (6.19a)$$

$$\mathbf{c}(v) = \left(\int_{A(v)} \tilde{\mathbf{K}}_{\mathbf{q}} \rho(\mathbf{q}) dF(\mathbf{q}) \right)^{-1} \int_{A(v)} \tilde{\mathbf{K}}_{\mathbf{q}} \mathbf{q} \rho(\mathbf{q}) dF(\mathbf{q}). \quad (6.19b)$$

Setting $d_{\mathbf{q}, \mathbf{p}_i}^p$ in Equation (6.3) to $\|\tilde{\mathbf{B}}_{\mathbf{q}}(\mathbf{p}_i - \mathbf{q})\|_2^2 = (\mathbf{p}_i - \mathbf{q})^T \tilde{\mathbf{K}}_{\mathbf{q}}(\mathbf{p}_i - \mathbf{q})$ and using Equation (6.18b) under the approximation that the metric tensor

remains constant over a given vertex area, $\tilde{\mathbf{K}}_{\mathbf{q}} = \tilde{\mathbf{K}}_{\mathbf{v}}$, leads to

$$\mathcal{H}_{\mathcal{G}}(\mathbf{P}_G) = \sum_{i=1}^n \sum_{v \in V_{G_i}} \left[J_{A(v), \mathbf{c}(v)} + M_{A(v)} (\mathbf{v}_{\mathbf{p}_i} - \mathbf{c}(v))^T \tilde{\mathbf{K}}_{\mathbf{v}} (\mathbf{v}_{\mathbf{p}_i} - \mathbf{c}(v)) \right]. \quad (6.20)$$

Expanding the expression $(\mathbf{v}_{\mathbf{p}_i} - \mathbf{c}(v))^T \tilde{\mathbf{K}}_{\mathbf{v}} (\mathbf{v}_{\mathbf{p}_i} - \mathbf{c}(v))$ gives us

$$\mathbf{c}(v)^T \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{c}(v) + \mathbf{v}_{\mathbf{p}_i}^T \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{v}_{\mathbf{p}_i} - 2 \mathbf{v}_{\mathbf{p}_i}^T \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{c}(v),$$

and by substituting we get from Equation (6.20)

$$\begin{aligned} \mathcal{H}_{\mathcal{G}}(\mathbf{P}_G) = & \sum_{i=1}^n \left[\sum_{v \in V_{G_i}} J_{A(v), \mathbf{c}(v)} + \sum_{v \in V_{G_i}} M_{A(v)} \left(\mathbf{c}(v)^T \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{c}(v) \right) + \right. \\ & \left. \mathbf{v}_{\mathbf{p}_i}^T \left(\sum_{v \in V_{G_i}} M_{A(v)} \tilde{\mathbf{K}}_{\mathbf{v}} \right) \mathbf{v}_{\mathbf{p}_i} - 2 \mathbf{v}_{\mathbf{p}_i}^T \left(\sum_{v \in V_{G_i}} M_{A(v)} \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{c}(v) \right) \right]. \end{aligned} \quad (6.21)$$

From Equation (6.21) follows that it is sufficient to minimize for the last two terms, since the first two terms do not depend on a particular choice of positions \mathbf{P}_G and remain unchanged with respect to the partition \mathcal{G} . The generator positions \mathbf{P}_G can now be chosen freely. Similar to Chapter 5, the generators can represent the real robot positions or the positions of virtual robots. If we use virtual generators and select the generators \mathbf{P}_G in Equation (6.21) to always be the centroids of the Voronoi regions, which is in accordance with the definition of a CVT, the last two terms in Equation (6.21) can be further simplified. The anisotropic centroids on the graph are obtained as

$$\mathbf{v}_{\mathbf{c}_i} = \left(\sum_{v \in V_{G_i}} M_{A(v)} \tilde{\mathbf{K}}_{\mathbf{v}} \right)^{-1} \left(\sum_{v \in V_{G_i}} M_{A(v)} \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{c}(v) \right), \quad (6.22)$$

which are the minimizers of the coverage cost. Substituting the real robot positions $\mathbf{v}_{\mathbf{p}_i}$ by the virtual generator positions $\mathbf{v}_{\mathbf{c}_i}$ for the last two terms of Equation (6.21) results in the partial coverage cost

$$\mathcal{H}_{\mathcal{G}}^*_{\text{aniso}}(\mathbf{P}_G) = \sum_{i=1}^n \left[- \mathbf{v}_{\mathbf{c}_i}^T \left(\sum_{v \in V_{G_i}} M_{A(v)} \tilde{\mathbf{K}}_{\mathbf{v}} \mathbf{c}(v) \right) \right], \quad (6.23)$$

which does not explicitly depend on the robot positions $\mathbf{v}_{\mathbf{p}_i}$ anymore. In the isotropic case, the centroids are given by

$$\mathbf{v}_{\mathbf{c}_i} = \frac{\sum_{v \in V_{G_i}} M_{A(v)} \mathbf{c}(v)}{\sum_{v \in V_{G_i}} M_{A(v)}}, \quad (6.24)$$

and Equation (6.23) can be reduced to

$$\mathcal{H}_{\mathcal{G}_{\text{iso}}}^*(\mathbf{P}_G) = \sum_{i=1}^n \left[- \frac{\left\| \sum_{v \in V_{G_i}} M_{A(v)} \mathbf{c}(v) \right\|_2^2}{\sum_{v \in V_{G_i}} M_{A(v)}} \right]. \quad (6.25)$$

The original problem of distributing n robots r_i over the surface \mathcal{S} can now be solved by simply minimizing Equation (6.23), or Equation (6.25) respectively, and letting the robots at positions \mathbf{p}_i approach the virtual generators at positions $\mathbf{v}_{\mathbf{c}_i}$ over time. As we will see in the following section, the partial cost $\mathcal{H}_{\mathcal{G}_{\text{aniso}}}^*(\mathbf{P}_G)$ and $\mathcal{H}_{\mathcal{G}_{\text{iso}}}^*(\mathbf{P}_G)$ are minimized in an efficient way by exchanging vertices between adjacent Voronoi regions, which only requires updates of the sums $\sum M_{A(v)} \mathbf{K}_{\mathbf{v}} \mathbf{c}(v)$ and $\sum M_{A(v)} \mathbf{K}_{\mathbf{v}}$, or $\sum M_{A(v)} \mathbf{c}(v)$ and $\sum M_{A(v)}$ respectively.

Coordination by Vertex Exchange

The second coverage solution computes an approximative DCVT on the graph and minimizes the coverage cost by exchanging boundary vertices iteratively across the boundaries of adjacent Voronoi regions. This *local vertex exchange* can be seen as a sequence of local optimization steps on the vertices at the boundaries of the Voronoi regions. Algorithm 8 describes the coverage algorithm of this second approach.

For each vertex v_A that lies at a boundary in a Voronoi region V_{G_i} , a local test is executed. If a vertex at the boundary next to v_A has not yet been assigned to a Voronoi region, i.e., the vertex is free, it is incorporated directly into V_{G_i} without any calculation of cost. If there exists at least one neighboring vertex $v_B \in V_{G_j}$, the change in the coverage cost $\mathcal{H}_G(\mathbf{P}_G)$ is calculated for the following three cases: (1) v_A is added to V_{G_j} , (2) v_B is added to V_{G_i} , or (3) no vertex is exchanged across the boundary (see Figure 6.3). The case resulting in the highest reduction of coverage cost is selected and the Voronoi regions are updated accordingly. This way, the Voronoi regions grow into areas of unassigned vertices while the ongoing exchange of vertices with other Voronoi regions helps to minimize the overall coverage cost.

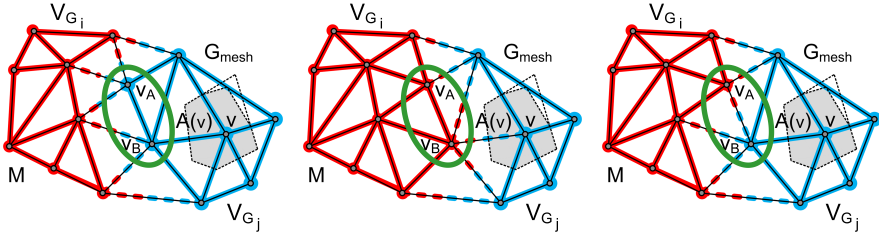


Figure 6.3: Vertex exchange. Left: Vertex v_A is added to the Voronoi region V_{G_j} (case (1)). Center: Vertex v_B is added to the Voronoi region V_{G_i} (case (2)). Right: No vertex is exchanged across the boundary of the Voronoi regions (case (3)).

Algorithm 8 Coordination by Vertex Exchange

Require: Voronoi region G_i , graph G_{M_i} and mesh M_i ; k_R rounds of vertex exchange.

- 1: $\{\mathcal{N}_i, G_{M_i}, M_i\} \leftarrow \text{UpdateNeighborhood}(\text{COM})$
 - 2: $G_i \leftarrow \text{AssignFreeVertices}(G_i, G_{M_i})$
 - 3: **for all** r_j with $j \in \mathcal{N}_i$ **do**
 - 4: $\text{ExchangeVertices}(G_i, G_j, k_R)$
 - 5: **end for**
 - 6: $\mathbf{v}_{g_i} \leftarrow \text{UpdateGoal}(G_i)$
 - 7: $\text{UpdateState}()$
-

Each robot r_i stores information about the vertices of its Voronoi region, such as the set of boundary vertices and the identity of the adjacent vertices outside the own Voronoi region, in memory. The update of the Voronoi regions and the DCVT can be realized efficiently through repeated evaluation and update of the sums in Equation (6.23) and Equation (6.25), respectively. Only a few addition and subtraction operations are required to correct the sums for the partial cost of an exchanged vertex.

If a vertex exchange leads to a disconnected region, the third case applies and the respective vertex will not be exchanged. Note that the third case does not affect the convergence of the overall algorithm; the coverage cost is not minimized but remains unchanged in this iteration step. However, over subsequent iterations, the pairwise optimization by the vertex exchanges among adjacent Voronoi regions minimizes the overall cost.

The underlying structure of the second coverage algorithm is similar to the one of the first coverage algorithm and builds on Algorithm 6. The robots

execute either a coordination or a moving action. Under the coordination action, first all the boundary vertices in V_{G_i} adjacent to free vertices in the graph G_{M_i} are updated. Robot r_i synchronizes its boundaries by requesting the identities of the boundary vertices of the neighboring robots. If the free vertices have not been occupied by another robot since last coordination, the vertices are added to V_{G_i} and its boundary is adjusted appropriately. The vertices that belong to a neighbor robot and adjoin V_{G_i} undergo the procedure of local vertex exchange. Robot r_i updates the information about the vertices according to the assignment and communicates these changes to the other robots. Hence, robot r_i requests data to perform the local optimization and sends the result back to each neighbor. While this bidirectional communication takes place, both robots are not allowed to answer another request.

k_R rounds of vertex exchange are performed by the `ExchangeVertices` function. With k_R chosen large enough, a robot will not enter a local vertex exchange with another robot neighbor before a partition with optimal cost between its own and the current neighboring region is obtained. Provided that a vertex exchange procedure can find the global optimum for the two regions, this leads to pairwise optimal partitions, similar to the method of Durham et al. (2012).

In situations, where a boundary vertex v_A of a Voronoi region V_{G_i} is connected to three or more vertices, which belong to Voronoi regions of different neighboring robots, additional communication is required. The local vertex exchange of vertices v_A and v_B between two adjacent robots and their Voronoi regions V_{G_i} and V_{G_j} does not influence a third robot's Voronoi region V_{G_k} ; but the set of boundary vertices stored by the third robot r_k needs to be adjusted when vertex v_A or vertex v_B changes the Voronoi region.

6.3.4 Properties of the Surface Coverage Algorithms

In the following, we further discuss some of the properties of the two presented coverage solutions.

Coupling of Coordination and Moving Actions

In the original Voronoi coverage method of Cortés et al. (2004), the robots act as the generators and converge to the mass centers of their Voronoi regions. However, this is no longer a necessary condition in our case, as the Voronoi regions are generated either by wavefront propagation between vertices or by local vertex exchanges of boundary vertices on the underlying graphs. The coordination and path planning actions are only loosely coupled. The

dependence is given through the communication and environment update. The robot positions finally determine which robot neighbors are in communication range and what the robots will perceive from their environment and incorporate into their mesh maps.

Similar to the differentiation between real and virtual generators and real and virtual goals in Chapter 5, we can introduce virtual and real positions into the mesh maps. The decoupling of the two actions allows to place the robots' goal positions temporarily at different locations in the Voronoi regions, which may improve the overall performance of the algorithms. Finally, the robots will converge to a CVT again. Depending on the task the robots need to perform, additional freedom in the goal placement can lead to lower cost paths and better accessibility of the goal positions, or better sensor coverage by adjusting the partition to local environment properties (Valette et al., 2008). Placing goals, or waypoints on the path toward a goal respectively, closer to the boundary of a Voronoi region can increase the explorative behavior of the basic coverage algorithms for example.

Related to the above discussion is how the two actions of coordination and moving are executed in relation to each other. Algorithm 6 can run in different variants: synchronous or asynchronous, sequentially or in parallel, or offline. In synchronous mode, the robots synchronize, wait until each robot has reached its centroid and update their Voronoi regions and regions' centroid all together. Asynchronous mode, as opposed to synchronous mode, means that the robots do not wait for the other robots at arrival at their centroid, and update their Voronoi regions and regions' centroids asynchronously. If the algorithm runs sequentially, the coordination and moving actions alternate. This particularly applies to a synchronized version of the algorithm. For increased efficiency, the algorithm is executed in parallel. The robots move and coordinate at the same time, which can still be realized as a synchronous or asynchronous process. If the robots are well informed, e.g., when the environment is known at start, the algorithm can run with a precomputed DCVT on the known graph, i.e., the coordination action is performed offline. In this case, each action is entered once: first the optimal partition is computed, then the robots deploy to their final goals at the mass centers of the Voronoi regions to form the centroidal Voronoi configuration.

Convergence of the Coverage Algorithms

In the following, we assume a finite environment and prove convergence of the two coverage algorithms in the case of known environments.

The proofs for a priori unknown environments are extensions, relying on the fact that each subgraph G_{M_i} , representing the mesh map M_i of robot r_i ,

changes only for a finite number of times given the underlying finite graph G_M . Convergence then results from the concatenation of a finite number of sequences. Changes in a subgraph G_{M_i} only occur at the switch of a sequence. During a sequence, the subgraphs remain unchanged and Proposition 6.1 for known environments applies.

Proposition 6.1. (Convergence of Wavefront Propagation) *A group of n robots r_i covers a graph G_M and converges to a centroidal Voronoi configuration with a local minimum in the coverage cost of Equation (6.4) by performing Algorithm 7.*

Proof. The shortest path distance $d_G(v, v_{\mathbf{p}_i})$ is strictly increasing on $\mathcal{S}_{v, v_{\mathbf{p}_i}}$, which is the vertex sequence given by the shortest path from v to $v_{\mathbf{p}_i}$ on the graph G_M . Therefore, the graph Voronoi partition \mathcal{G} minimizes $\mathcal{H}_G(\mathbf{P}_G)$ for any fixed robot configuration $\mathbf{P}_G = [\mathbf{v}_{\mathbf{p}_i}]_{i=1}^n$, i.e., $\mathcal{H}_G(\mathbf{P}_G, \mathcal{G}(\mathbf{P}_G)) \leq \mathcal{H}_G(\mathbf{P}_G, \mathcal{Y})$, where \mathcal{Y} is an arbitrary partition of the graph. Let T be the mapping from vertices \mathbf{P}_G to goal vertices $\mathbf{P}_G^* = [\mathbf{v}_{\mathbf{p}_i}^*]_{i=1}^n$, which is the vector formed by the positions of the neighboring vertices $v_{n_{il}}$ of $v_{\mathbf{p}_i}$ that lie on the shortest paths to the centroids $v_{\mathbf{c}_i}$, with $i \in \{1, \dots, n\}$, $l \in \mathbb{N}_{>0}$. We get in synchronous mode $T : [\mathbf{v}_{\mathbf{p}_1}, \dots, \mathbf{v}_{\mathbf{p}_i}, \dots, \mathbf{v}_{\mathbf{p}_n}] \mapsto [\mathbf{v}_{\mathbf{p}_1}^*, \dots, \mathbf{v}_{\mathbf{p}_i}^*, \dots, \mathbf{v}_{\mathbf{p}_n}^*]$, and in asynchronous mode $T : [\mathbf{v}_{\mathbf{p}_1}, \dots, \mathbf{v}_{\mathbf{p}_i}, \dots, \mathbf{v}_{\mathbf{p}_n}] \mapsto [\mathbf{v}_{\mathbf{p}_1}, \dots, \mathbf{v}_{\mathbf{p}_i}^*, \dots, \mathbf{v}_{\mathbf{p}_n}]$. The mapping T has the property $\mathcal{H}_G(T(\mathbf{P}_G), \mathcal{Y}) \leq \mathcal{H}_G(\mathbf{P}_G, \mathcal{Y})$, which is guaranteed by Algorithm 7, which requires $h(\mathbf{v}_{\mathbf{p}_i}^*, V_{G_i}) < h(\mathbf{v}_{\mathbf{p}_i}, V_{G_i})$ for each robot r_i . Inequality $\mathcal{H}_G(T(\mathbf{P}_G), \mathcal{G}(T(\mathbf{P}_G))) \leq \mathcal{H}_G(T(\mathbf{P}_G), \mathcal{G}(\mathbf{P}_G))$ follows directly from the optimality of the Voronoi tessellation for a fixed set of points. Since the property of T holds for an arbitrary tessellation \mathcal{Y} , we finally get with $\mathcal{Y} = \mathcal{G}(\mathbf{P}_G)$: $\mathcal{H}_G(T(\mathbf{P}_G), \mathcal{G}(T(\mathbf{P}_G))) \leq \mathcal{H}_G(T(\mathbf{P}_G), \mathcal{G}(\mathbf{P}_G)) \leq \mathcal{H}_G(\mathbf{P}_G, \mathcal{G}(\mathbf{P}_G))$, and thus the cost is minimized in each iteration step of Algorithm 7. \square

Proposition 6.2. (Convergence of Vertex Exchange) *Given the connectivity of the vertices of the single Voronoi regions V_{G_i} , a group of n robots r_i covers a graph G_M and converges to a centroidal Voronoi configuration with a local minimum in the coverage cost of Equation (6.20) by performing Algorithm 8.*

Proof. The Voronoi regions V_{G_i} grow to cover the free vertices of graph G_M until full coverage of the graph is reached, i.e., $\mathcal{H}_G(\mathbf{P}_G)$ increases constantly. An upper bound in the cost is however given by the finite size of G_M . No other increase of the cost is induced. In order to maintain the connectivity of the Voronoi regions, vertices may not be exchanged, i.e., further minimization is suppressed but the coverage cost is not increased either. $\mathcal{H}_G(\mathbf{P}_G)$ is a

positive value and permanent exchanges of vertices across the boundaries of V_{G_i} reduce the cost locally in each iteration step, in synchronous as well as asynchronous mode. A point is reached where no vertices can be exchanged to further decrease the cost and Algorithm 8 has converged to a configuration of local minimum cost. As each robot r_i is moved continuously to the centroid of its Voronoi region by Algorithm 8, and the minimization of Equation (6.20) implies a Voronoi tessellation, convergence to a centroidal Voronoi configuration follows. \square

The two coverage solutions converge to a centroidal Voronoi configuration, and convergence can be used as criterion to stop the deployment state. In the context of hybrid coverage (see also Chapter 4 and Section 6.4 below), convergence triggers the switch to the sweeping state of the second stage.

Arbitrary p -Norms for the Second Approach (Algorithm 8)

The p -norm only applies to the first approach, which computes geodesic distances. Note that the second approach with approximate distance computation, as presented here, requires the 2-norm and does not hold for p -norms in general. This is inherent to above derivation from Equation (6.3) to Equation (6.21), which is based on the parallel axis theorem. The parallel axis theorem relies on specific properties of the 2-norm and does not in general apply for arbitrary p -norms. We refer to Appendix A for a formal proof.

We further observe that, by a variation of the transformation that leads to Equation (6.21), i.e., by using the polynomial from Proposition A.1 instead of the parallel axis theorem, we can obtain a new (more complex) expression, which replaces Equation (6.21). This allows again for efficient cost computation and cost update in a similar way to Equation (6.21). Furthermore, a similar transformation could alternatively be derived by using a generalization of the parallel axis theorem to arbitrary p -norms that is different from the generalization assumed by Corollary A.2.

6.4 Extensions to Adaptive and Hybrid Coverage Control

In this section, we present two extensions of the proposed surface coverage algorithms. By making use of the metric tensor field, which is included in the distance computation of the two coverage algorithms, we can incorporate additional adaptivity and user guidance into the coverage solutions. Another

extension applies the coverage algorithms under the hybrid coverage concept of Chapter 4 for incremental surface area coverage, using hierarchical composition.

6.4.1 Adaptive Surface Coverage

Voronoi coverage methods partition the environment and assign robots to Voronoi regions. Each robot typically completes a task within its region. For example, the robots monitor their regions, provide services to locations within their regions, or sweep the areas of their regions entirely. The size, position and shape of a Voronoi region influences the robot's efficiency in completing such a task. The added adaptivity in the surface coverage algorithms can help to shape the robots' regions with respect to the environment and tasks to be performed by the robots.

The Voronoi regions adapt to local anisotropy, which is defined by a metric tensor field on the curved surface. The metric tensor field allows for controlling shape, density and size, as well as orientation and aspect ratio of the Voronoi regions. This new adaptivity may improve multi-robot coverage in several ways. First, adapting the size and orientation of the Voronoi regions according to environment characteristics like surface curvature, salient features or representation uncertainty makes robot movements during task completion in a region on the surface safer and more efficient. Second, adapting the density, orientation or aspect ratio of the Voronoi regions by an input tensor field enables user guidance of the robot configuration. Finally, adapting the shape and size of the Voronoi regions allows to match the region to a sweeping pattern, which is executed by a robot in the region. This corresponds to the second stage in the hybrid coverage concept (see Section 4.2 and Section 6.4.2 below).

The Finsler tensor field given by \mathbf{B}_v offers several ways to adapt the partition over the surface \mathcal{S} (see Figure 6.9 for an example). The orientation of the Voronoi regions is influenced by the directions of the eigenvectors of \mathbf{B}_v . The aspect ratio of the regions is given by the ratios of the eigenvalues of \mathbf{B}_v , measuring the strength of directionality. The size of the regions can be changed by the weighting factor or mass density ρ . In addition, the distance in the coverage cost can assume the general p -norm for the first approach. Depending on the selection of p , the p -norm results in a more circular or square-shaped distance field, which leads to additional modifications in the shape of the partitions.

6.4.2 Application to Hybrid Coverage

Next we discuss how the surface coverage algorithms with added adaptivity can be applied within the hybrid coverage concept of Chapter 4 to achieve area coverage on a curved surface. The hybrid area coverage method is outlined in Figure 6.4 and Algorithm 9. The idea behind hybrid coverage is to combine robot deployment and sweeping motions. The method starts with the robots spreading out on the surface. The robots cooperatively partition the surface and get their assigned areas of operation. In this first stage, the surface coverage algorithms from Section 6.3, extended with adaptivity as mentioned above, are applied to realize an effective deployment and adaptive decomposition of the surface. The included adaptivity can be actively used to shape the Voronoi regions and simplify the coverage of the area, e.g., leading to compact or more elongated shapes that support circular spiraling or rectangular back-and-forth sweeping patterns. Upon convergence, the robots switch to the second stage, where each robot sweeps its assigned region locally to search the area.

Depending on the size of the environment to be covered and the range of coverage of the single robots, the two stages of deployment and sweeping are iterated. This corresponds to the hierarchical coverage solution suggested in Chapter 4. The robots relocate and redistribute outside the already covered area by applying the hybrid coverage subroutines `AdaptiveSurfaceCoverage` and `SweepSurfaceCoverage` again. By iterating the process, the complete surface is finally covered by the robots.

Once a Voronoi region is covered, it is marked as covered in the robot's mesh map M_i and is locked; the robot communicates the status to its neighbors, which update their mesh maps accordingly. As the surface coverage algorithms deploy the robots by generating a Voronoi tessellation, a dual Delaunay graph is established simultaneously (see Figure 6.9, Figure 6.10 and Figure 6.11 for examples). The graph connects the Voronoi regions, represents the surface topology and gives a simplified low resolution representation of the environment. This representation remains valid and may serve as roadmap for future relocation and redistribution phases of the robots. The covered regions are known to the robots, since they have swept and explored these areas before. Hence, robot paths transferring from one location on the surface to another are preferably planned through the known and safe area of the regions along the Delaunay graph.

Besides covering and locking of regions, many more operations on the regions are possible. A robot can leave its region uncovered and reassign it to other robots for coverage, or ask other robots for support. Regions can be deleted cooperatively if a robot fails or relocates. Furthermore, a new

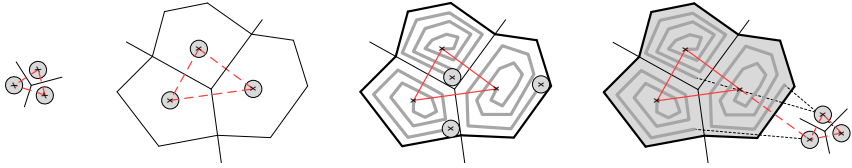


Figure 6.4: Hybrid surface area coverage. Left: Robot deployment. Center: Sweeping motions. Right: Relocation and redistribution. The dual Delaunay graph, which is visualized in red, is created simultaneously during the coverage process.

Algorithm 9 Hybrid Surface Area Coverage

Require: Set of n robots r_i , with sensing and communication capabilities.
 Subroutines for the two stages of deployment and sweeping motion.

- 1: **loop**
 - 2: AdaptiveSurfaceCoverage(state) // state == DEPLOY
 - 3: SweepSurfaceCoverage(state) // state == SWEEP
 - 4: RelocateAndRestart()
 - 5: **end loop**
-

region is instantiated whenever a robot joins during deployment, or a region is created inside already covered area in order to initiate redundant coverage.

6.5 Results

We have tested both of the proposed coverage solutions thoroughly. Simulations on different synthetic 3D mesh models with varying numbers of robots verify basic as well as extended functionalities of the two surface coverage algorithms. Experiments with a group of five e-puck robots were conducted on a test setup with curved surface to further evaluate the applicability of the solutions to real robot platforms.

We assume that the triangle mesh has been generated in advance and is available as input to the coverage algorithms. We investigate the algorithms under two variable assumptions: the robots operate in synchronous or asynchronous mode, and the environment may be known or unknown to the robots. If the environment is known, the robots know the entire mesh model a priori. In an unknown environment, the robots are able to sense the surface and discover mesh vertices within a sensor range of R_{sens} as they move along the triangle mesh. For each robot r_i , the detected vertices are

added to the already discovered subgraph $G_{M_i} \subset G_M$. The robots exchange information among each other within a communication range R_{com} . If two robots are within R_{com} and share at least one common vertex, i.e., their subgraphs G_{M_i} and G_{M_j} are connected, the vertices are exchanged and the subgraphs merged.

6.5.1 Comparison of Surface Coverage Algorithms

We first present simulations and experiments which analyze the basic operation of the two surface coverage algorithms. The surfaces are represented by regular triangle meshes of varying resolution. For the algorithm comparison, we use the 2-norm and uniform metrics in the algorithms. The first coverage solution is an implementation of Algorithm 6 and Algorithm 7 with the `UpdateGoalDirection` function implemented after Equation (6.16). As graph-based representation, we use the graph $G_M = G_{\text{mesh}}$. The second coverage solution implements Algorithm 6 and Algorithm 8 with $k_R = 1$ rounds and the `ExchangeVertices` function realized after Equation (6.25). Here, the mesh is represented by the dual graph, $G_M = G_{\text{mesh}}^*$.

Simulation Results

We use Matlab as simulation environment. The two coverage solutions were tested on different standard 3D mesh models from computer graphics. In addition, the triangle mesh from the bumpy slope test setup was used for simulations. The robots are holonomic point robots.

Varying Geometric Complexity. Simulations on standard 3D mesh models, such as the Stanford Bunny (see Figure 7.2 for the model), or basic geometric shapes like torus and sphere, demonstrate that the coverage solutions can cover arbitrary curved surfaces in 3D space. The resulting robot deployments were evaluated qualitatively. The evaluations included variations of the robots' initial positions for a given mesh model and tests with all four combinations of the algorithm settings: synchronous vs. asynchronous mode, known vs. unknown underlying mesh model.

Varying Number of Robots. Simulations were run for different numbers of robots. 20 simulation runs were executed per group, with group sizes of 5, 10 and 20 robots. The mesh of the bumpy slope with 50 mm edge length was used. The algorithms were set to synchronous mode and the environment was known. The initial configuration of the robots on the mesh was selected for each run from the entire mesh uniformly at random. Thus, the simulations converged to different local minima. The graphs with the plots of all 20

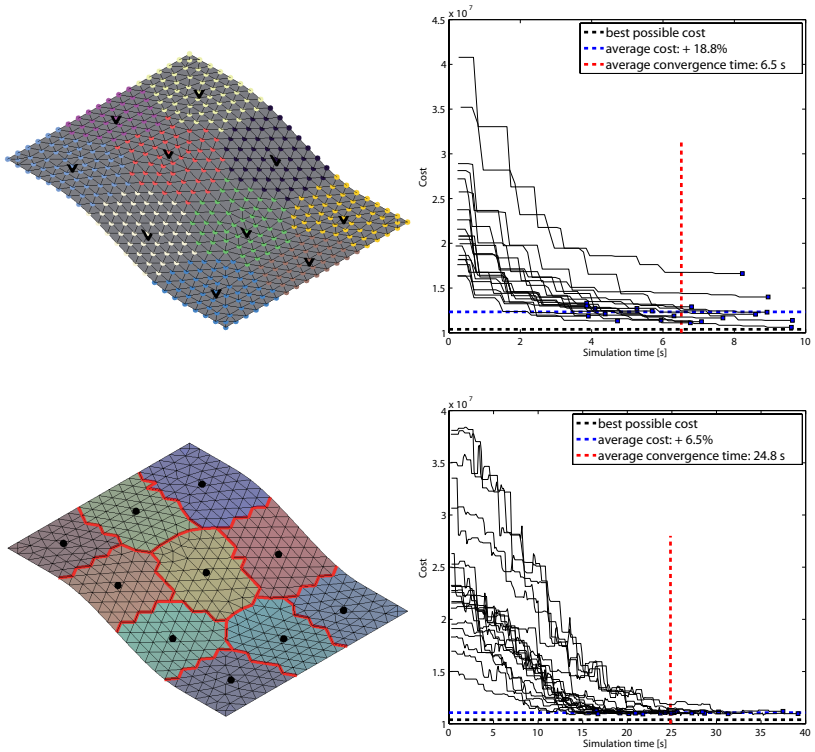


Figure 6.5: Simulations on a curved surface. A group of 10 robots covers the surface mesh of the bumpy slope. The best final centroidal Voronoi configurations (left column) and the cost from 20 simulation runs (right column) are shown for Algorithm 7 (top row) and Algorithm 8 (bottom row). The best possible cost, average final cost and average convergence time are indicated in the plots.

simulations (black lines) with their final values (blue squares) are visualized for the group with 10 robots for both coverage solutions in Figure 6.5. The average convergence time is 6.5 s for the first solution and 24.8 s for second solution (red dashed line). The average final coverage cost (blue dashed line) for the first solution is 18.8 % and for the second solution 6.5 % above the best possible final coverage cost (black dashed line). The best possible final coverage cost approximates the global minimum of \mathcal{H}_G and is computed as an upper bound by the Lagrangean relaxation heuristics according to Beasley (1993).

The best possible coverage cost at final configuration decreases due to the summation over smaller and smaller distances with increasing number of robots—for 5 robots: $2.23 \cdot 10^7$, 10 robots: $1.45 \cdot 10^7$, 20 robots: $0.52 \cdot 10^7$. On the other hand, we found that the average deviation in coverage cost increases approximately linear in the number of robots. A reason for this effect is the increasing robot density on the graph, which causes the robots to block each other when badly initialized.

Note that the cost functions of the first and second coverage algorithms cannot be compared directly, since the coverage cost in Equation (6.25) is partial and approximative, and the Voronoi regions initially grow during execution of the second algorithm. Therefore the cost of the second algorithm is recalculated for evaluation purposes based on the latest robot positions in each time step by using Equation (6.4), which is similar to the way it is done in the first algorithm.

Experimental Results

We use five e-puck robots and the bumpy slope test setup, with and without additional obstacle, for the experiments. Further details about the robots and the test setup are provided in Chapter 2.

Mesh Resolution. The performance of the first and second coverage solution is analyzed for different sizes of the triangles in the mesh. Meshes with triangles of edge lengths of 50 mm, 100 mm and 200 mm are tested. Meshes with edge lengths of 200 mm are too coarse for the first solution, and the robots get stuck in a suboptimal local minimum. The second solution is more robust against varying triangle sizes and also works for edge lengths of 200 mm meshes. Basically, coarser meshes have the advantage of faster convergence as the number of vertices, and thus the computational cost, are greatly reduced. Additionally, the robots move over longer distances along an edge or face until they reach a next coordination action where they reorient before driving straight again. The reduction in required coordination actions and thus in robot rotations saves additional time. Convergence time grows roughly by a factor of 1.5 at each transition, from a mesh with 200 mm edge length to a mesh with 100 mm edge length, and from a mesh with 100 mm edge length to a mesh with 50 mm edge length. For our setup, the mesh with 100 mm edge length resulted in fast convergence to a well-balanced centroidal Voronoi configuration in simulation as well as in experiments. The number of vertices is ideally as low as possible but without losing details of the structure due to a low mesh resolution.

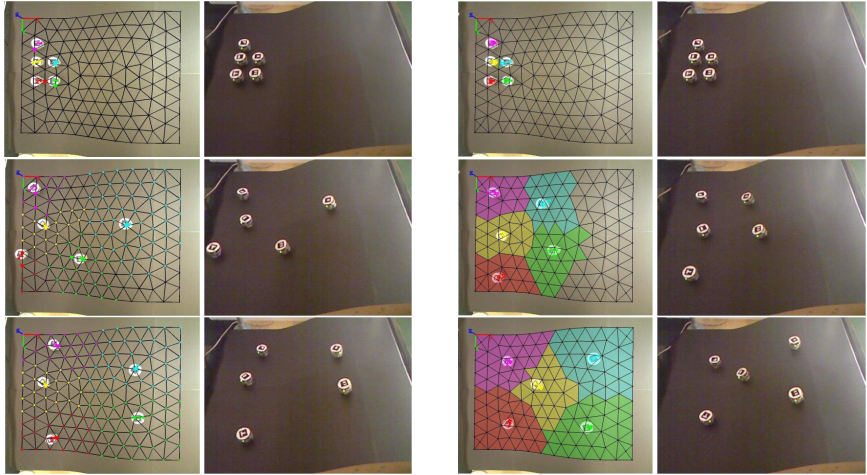


Figure 6.6: Experiments on a curved surface. A group of five e-pucks deploys according to Algorithm 7 (left) and Algorithm 8 (right) for asynchronous mode and unknown environment. The views from the overhead camera show for both algorithms the initial, intermediate and final configurations during coverage of the surface (the views are overlaid with the surface mesh and the Voronoi partition; Voronoi regions are visualized in the colors of the robots).

Coupling of Actions: Synchronous vs. Asynchronous Mode. Experiments confirm that the robots converge faster to the final configuration in asynchronous mode than in synchronous mode. As expected, the robots do not have to wait for the other robots when computing their next goal positions, which speeds up the robot deployment. The time difference is more distinct for the second coverage solution, since the robots drive along several triangle centroids until they reach their next goal positions, i.e., a sequence of the moving action may take longer than for the first coverage solution. This results in longer waiting times in the synchronous mode.

Convergence and Repeatability. In total, we ran over 50 experiments on the bumpy slope test setup for each of the two coverage solutions. Thereof, 25 experimental runs were performed in asynchronous mode and for unknown environment, which is the setting that is the most desirable for real applications. A triangle mesh with 100 mm edge length was used. The sensor range R_{sens} was set to 0.5 m and the communication range R_{com} was kept infinite. A sequence from an experimental run is shown in Figure 6.6 for both of the

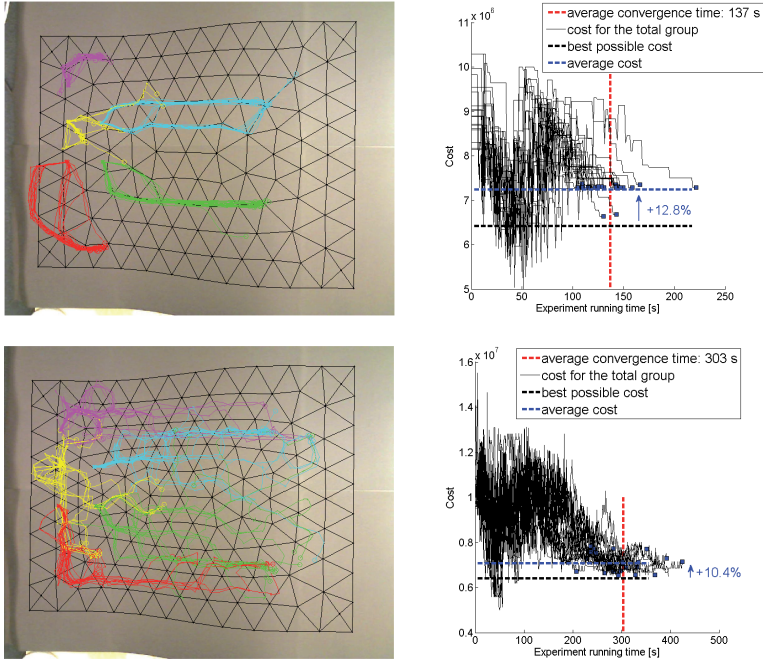


Figure 6.7: Statistical evaluation. 25 experimental runs are evaluated on the curved surface with Algorithm 7 (top row) and Algorithm 8 (bottom row) for the group of five e-pucks in asynchronous mode and unknown environment. Left: Trajectories of the robots from 25 deployments on the surface mesh. Right: Coverage cost for the whole group, plotted over the running time. The best possible cost, average final cost and average convergence time are indicated in the plots.

two solutions. The initial configuration of the five robots is shown in the top row of the figure.

Over the 25 experimental runs, the first solution results in an average convergence time of 137 s with a standard deviation of 28.6 s (Figure 6.7, top right). The second solution takes longer to converge than the first solution, and evaluations show an average convergence time of 303 s with standard deviation of 55.8 s (Figure 6.7, bottom right). The lower convergence rate is in accordance with the simulation results. There are two main reasons. First, in our implementation the robots move from triangle centroid to triangle centroid on straight lines, which leads to zig-zag paths and the robots rotate more often than for the first solution. Second and more interestingly, as the

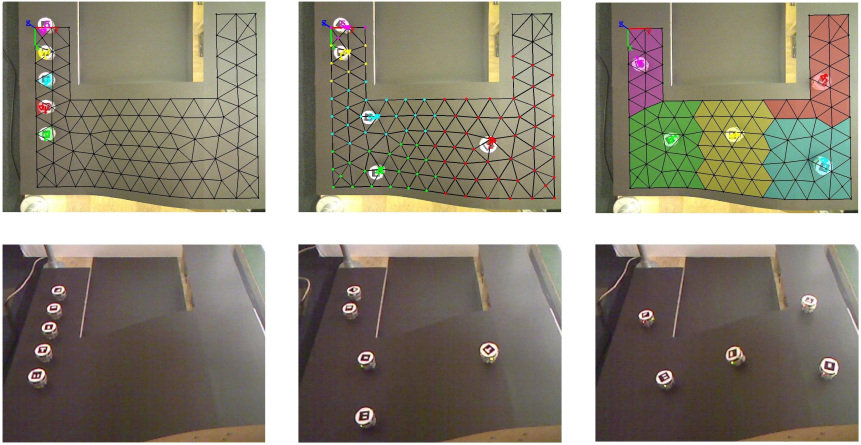


Figure 6.8: Additional nonconvexity. Left: Initial robot configuration on a curved surface with a hole. Center: Suboptimal distribution of the five e-pucks under Algorithm 7. The robots get stuck in a suboptimal local minimum when starting from the initial configuration. Right: Under Algorithm 8, the five e-puck robots succeed in covering the surface with the hole.

Voronoi regions grow vertex by vertex, they are created slower compared to the first solution.

The first solution mostly converges to the same final configuration over the 25 experimental runs for asynchronous mode and unknown environment (Figure 6.7, top left). The final configuration for the first solution, however, strongly depends on the initial configuration. Results from experiments with the second solution show that the robots distribute into different final configurations (Figure 6.7, bottom left). The order in which the boundaries of the Voronoi regions are updated influences the vertex exchanges and the Voronoi regions evolve differently from run to run. Thereby the second solution reaches slightly lower values in coverage cost compared to the first solution. The average cost of the final configuration for the second solution lies 10.4 % above the best possible coverage cost, whereas the average cost of the first solution is 12.8 % above. Besides, the final coverage cost of the second solution varies in a larger range since different minima of the cost are reached by the final configurations.

Additional Nonconvexity. We also tested the two coverage algorithms for different test setups. In this experiment, the middle part was removed to add a hole obstacle to the curved surface. The algorithms now must deal with an additional nonconvexity. The resulting coverage of the surface is shown in Figure 6.8 on a mesh with 100 mm edge length for both coverage solutions. The algorithms are set to asynchronous mode and the environment is unknown. Whereas the second solution generates well-balanced Voronoi regions, the first solution converges to a local minimum with suboptimal distribution of the robots on the surface. A reason for the suboptimal coverage is the first solution’s strong dependence on the initial configuration. In this particular case all the robots are lined up one after the other on the narrow branch. The unfavorable start positions lead to the suboptimal coverage. Tests on a mesh with 50 mm edge length showed that the use of a denser mesh improves the final partition. Besides, additional weights in the graphs can further improve the partitions.

6.5.2 Extensions of Surface Coverage Algorithms

The adaptive versions of the surface coverage algorithms and the algorithm for hybrid surface area coverage were tested in simulation for different surfaces. The algorithms are implemented in C++ and ROS. The surface models vary in complexity and consist of purely synthetic meshes as well as meshes obtained by surface reconstruction from real laser point clouds.

Adaptive Surface Coverage

We give two examples for adaptive surface coverage in the following. We use the second coverage algorithm with the `ExchangeVertices` function now implemented after Equation (6.23), and operate on the graph $G_M = G_{\text{mesh}}$ instead of using the dual graph.

The first experiment demonstrates the adaptivity added to the Voronoi coverage. Figure 6.9 shows the deployment of four robots on a sphere under varying user guidance. The metric tensor field for controlling the adaptivity is specified directly by the user. The first deployment is uniform, which sets equal weights to all vertices and directions. The second deployment is isotropic and directs the robots onto a great circle of the sphere. The metric tensor field for the first two deployments is of the form $\tilde{\mathbf{K}}_{\mathbf{v}} = \mathbf{I}_3$, and $M_{A(v)}$ is constant for the uniform deployment and varies over the sphere as a function of location for the isotropic deployment. The third and fourth deployments use an anisotropic metric. The Voronoi partitions point along the horizontal and vertical directions.

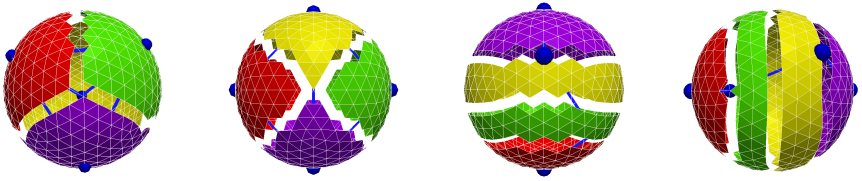


Figure 6.9: Adaptive surface coverage of a sphere. Adaptive deployment of four robots (blue balls) on a sphere. From left to right: Uniform, isotropic (weights placed toward the great circle) and anisotropic (horizontal and vertical) metric tensor fields provided by the user guidance. The dual Delaunay graph (blue lines), which connects the Voronoi regions, is simultaneously created.

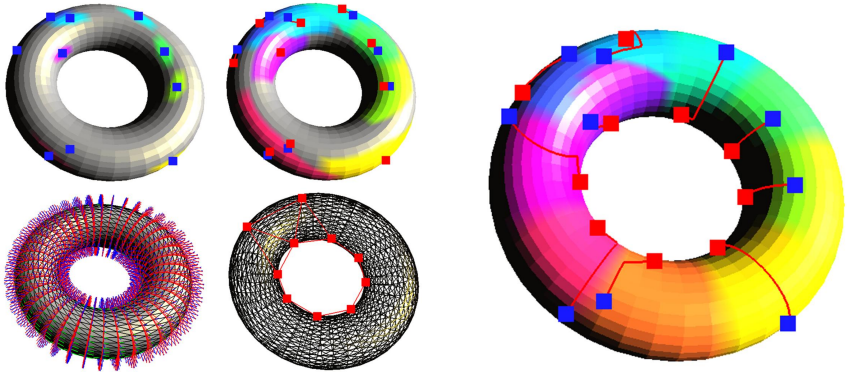


Figure 6.10: Adaptive surface coverage of a torus. Adaptive deployment of 10 robots (red cubes), starting from their initial configuration (blue cubes), using an isotropic curvature metric. The principal curvature directions estimated for the torus are shown on the bottom left. The dual Delaunay graph (red lines) on the torus at convergence is visualized in the center.

In the second experiment, 10 robots are distributed on the synthetic mesh model of a torus (Figure 6.10). The deployment on the torus is again isotropic but the matrix $\tilde{\mathbf{K}}_{\mathbf{v}}$ is now constructed from the local curvature of the surface, given by the weights $\rho(\mathbf{v}) = 1 + \sqrt{k_{\mathbf{v},1}^2 + k_{\mathbf{v},2}^2}$, the principal curvatures $k_{\mathbf{v},1}$ and $k_{\mathbf{v},2}$, and the principal directions. The principal curvatures and principal directions are estimated from the mesh model following the approach by Meyer et al. (2002), and are shown in Figure 6.10 on the bottom left.

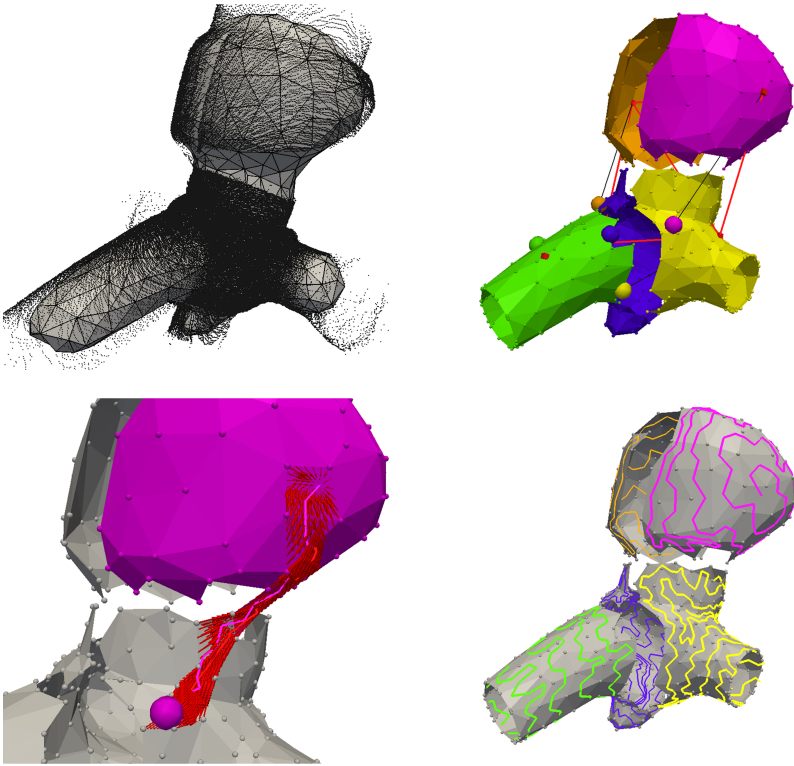


Figure 6.11: Hybrid surface area coverage of a steam chest. Top left: 3D laser point cloud and reconstructed triangle mesh of a steam chest tube. Top right: Voronoi partition resulting from the deployment of five robots (colored balls). Bottom left: Transitioning triangle strip path and vector field generated along the strip for controlling a robot toward the centroid of its Voronoi region. Bottom right: Covering triangle strip paths, defining a sweeping pattern for each of the Voronoi regions.

Hybrid Surface Area Coverage for Multi-Robot Inspection

In this last simulation experiment, the inner surface of the steam chest (see Section 2.1) is covered by five robots, applying the hybrid coverage concept. Figure 6.11 shows one iteration of the hybrid surface area coverage algorithm according to Algorithm 9. The input is a 3D point cloud, which was recorded with the rotating Hokuyo URG-04LX laser range finder (Figure 6.11, top

left). From the point cloud, a triangle mesh is reconstructed. Five robots are deployed over the mesh in a first stage, using the adaptive surface coverage algorithm (Figure 6.11, top right). In a second stage, each robot covers its Voronoi region with a sweeping motion (Figure 6.11, bottom right). The path planner computes transitioning as well as sweeping paths. Paths to transfer a robot to a goal position are planned by an A* search, which generates a triangle strip path on the mesh. The sweeping paths to cover the surface area result from a covering strip planner and the robots are controlled by smooth vector fields generated along the strip (the mesh-based path planners are the topic of the following Chapter 7).

6.6 Summary

The focus of this chapter is on the deployment of multiple robots to achieve Voronoi coverage on curved surfaces. The robot deployment can be seen as the first stage of a two-tired hybrid surface area coverage method. In this context, it is particularly important to provide an effective partitioning of the surface in the first stage, since each robot is required to sweep the entire area of its assigned Voronoi region in the subsequent second stage. Distributed multi-robot coverage of curved surfaces is useful for a broad class of real-world applications, such as collaborative operations in rough terrain or in inspection scenarios for industrial structures.

We present two adaptive surface coverage algorithms which deploy a group of robots on a graph and allow for adapting the size, positions and shapes of Voronoi regions by a metric tensor field defined on the surface. The tensor field can be computed from local surface properties or is controlled externally by user guidance. The first coverage solution propagates a discrete wavefront to construct a DCVT on the graph and computes gradients of the coverage cost locally to update the robots' goal directions. The second coverage solution constructs the DCVT by a local exchange of graph vertices and projects the computed goal positions from the ambient space back onto the graph. The two coverage solutions have been tested successfully in simulations and in experiments with real robots on different synthetic as well as more realistic mesh models. The analysis reveals some complementary trends in the characteristics of the two solutions, which provide interesting directions for future research.

The two coverage solutions build on a discrete environment representation. A graph is an abstract representation that provides a good choice. We use a triangle mesh to represent the curved surface in our implementation. However, a graph can be implemented more generally by any polygonal

mesh or grid, and works on surfaces as well as in full 2D, 3D or higher dimensional spaces. Coverage on a graph results in approximative solutions but resolves problems arising from nonconvexity or topological constraints, which are hard to handle otherwise. For example, the coverage problem of Chapter 5, where geometric constraints are imposed by a nonconvex planar environment, can alternatively be solved by the two coverage algorithms presented in this chapter. The advantage of a graph-based representation is that a graph may only model the feasible set, which is intrinsically unconstrained. Handling free-standing obstacles and moving on a curved surface then become related problems.

Furthermore, a discrete representation may not need to be a graph. Point clouds present an interesting alternative. The first coverage solution could operate on a neighborhood graph that is constructed locally from the point cloud (Mémoli and Sapiro, 2005; Tenenbaum et al., 2000). The second coverage solution even extends to point clouds without using any underlying explicit graph. Single points instead of vertices are exchanged locally across the Voronoi regions; only a neighborhood relation, e.g., given by an efficient nearest neighbor search, is required to maintain connectivity. This bears further interesting relations to the clustering of data points (Du et al., 1999).

Chapter 7

Mesh-Based Path Planning on Curved Surfaces

The motivation behind this and the next chapter is to provide navigation solutions for moving mobile robots on curved surfaces. The last chapter has proposed multi-robot coverage solutions that deploy robots on triangle meshes. For this method to work, an efficient implementation of the moving action is needed, such that the robots transition from their current state to the next goal state along feasible and preferably optimal trajectories on the surface. We present in the following a hierarchical navigation solution that plans a discrete triangle strip path on the mesh representation and applies continuous feedback control to guide a robot smoothly along the triangle strip to the goal regions on the surface.

Our key idea is to make use of *triangle strips*. First, triangle strips are related to computer graphics, where the stripification of a triangle mesh into single triangle strips is popular for data compression and efficient rendering of mesh models. In this context, many methods were developed which provide a solid basis for further studies. Second, such stripification can be seen as discrete path planning on the mesh. Triangle strips are extracted and the mesh is reduced this way to a representation that contains all the relevant information for path planning and robot control. Third, concerning multi-robot coverage, triangle strip planning allows not only to plan transitioning but also covering triangle strip paths, which define complete sweeping patterns over a surface. Finally, triangle strip planning relates to feedback motion planning and the composition of funnels paradigm (LaValle, 2006). The individual triangles of the triangle strips represent basic cells that cover the surface; navigation functions and vector fields can now be designed within

each triangle of the strip in order to move a robot from triangle to triangle, i.e., the states transition from funnel to funnel.

We implement a version of the triangle strip planner for navigation on curved surfaces, using and combining existing concepts from surface reconstruction and path planning. The chapter mainly contributes by investigating the potential applicability of these concepts within the realistic scenario of robotic inspection with climbing robots like MagneBike.

The navigation solution for mesh-based path planning with application to robotic inspection on curved surfaces of industrial structures has been presented at the International Conference on Applied Robotics for the Power Industry (Breitenmoser and Siegwart, 2012). We give an overview of related methods from surface reconstruction and graph-based path planning in Section 7.1. Section 7.2 describes the navigation solution, including environment representation, triangle strip planning and feedback control. An evaluation and discussion of the solution is presented in Section 7.3. A final summary concludes the chapter in Section 7.4.

7.1 Related Work

Our navigation solution relies on 3D point clouds recorded by a rotating laser range finder for the reconstruction of a navigable triangle mesh. Environment modeling from LIDAR data as well as path planning and control using reconstructed triangle meshes present a challenging problem. We review related work that addresses some of these challenges.

Typical robotic mapping methods are designed for the construction of grid maps of 2D and 2.5D terrains (Hebert et al., 1989; Moravec, 1996; Triebel et al., 2006). We are however interested in curved surfaces and full 3D environments and focus on the environment representation by triangle meshes in the following. Triangle meshes can be of advantage for several reasons: a mesh provides a compact data structure and represents a graph in 3D space, which can conveniently be used as representation for graph-based path planning algorithms. Furthermore, with respect to the inspection task, meshes can be used to map defects found during coverage of the surface, for logging already covered trajectories, or for visualization purposes in the interaction with human inspectors.

There exists a great variety of surface reconstruction methods in computer graphics literature (Alexa et al., 2001; Amenta et al., 2001; Carr et al., 2001; Dey and Goswami, 2004; Gopi and Krishnan, 2002; Hoppe et al., 1992; Kazhdan et al., 2006). However, it is often not clear which among those can be expected to perform well in a given environment or application. Another

challenge is due to the fact that surface reconstruction methods are generally developed for computer graphics applications with their main focus on the user-guided generation of high accuracy meshes for 3D modeling. Often less attention has been paid in the mesh generation to aspects important for mobile robotics like full automation, robustness against noise, dynamic mesh updates, real-time processing of sensor inputs, or limited resources (e.g., memory and computational power). In contrast, there are approaches emerging from robotics and graphics to close these gaps (Gingras et al., 2010; Mar-ton et al., 2009; Newcombe et al., 2011).

Stripification of triangle meshes is another broad topic in computer graphics because of its importance for 3D rendering. Our main interest is in single-strip representations. Diaz-Gutierrez et al. (2005) generates single triangle strips that satisfy additional constraints for high performance rendering. Gopi (2004) grows a single triangle strip incrementally without the need for preprocessing. The work by Speckmann and Snoeyink (2001) suggests the generation of triangle strips with the help of spanning trees. As shown by these works, the stripification of a triangle mesh relates to spanning trees (Agmon et al., 2006; LaValle, 2006) and space filling curves or trees (Kuffner and LaValle, 2011), which result in the complete coverage of a surface area.

Our original intention is path planning and coverage path planning on triangle meshes. In this context, the stripification can be seen as nothing else but the planning of discrete paths over a triangle mesh. Alternative approaches for graph-based path planning on the mesh involve classical graph search methods like A* or Dijkstra's algorithms. Enhanced algorithms, such as the methods of Koenig and Likhachev (2002) and Ferguson and Stentz (2006), offer additional properties like efficient replanning or improved path continuity.

A specific approach of feedback motion planning for navigating a robot in a planar polygonal environment is to first plan a discrete path and then to construct a vector field along the path that leads toward the goal (Belta et al., 2005; Conner et al., 2003; Lindemann and LaValle, 2005). The methods presented by Pimenta et al. (2007) and Pereira et al. (2009) use planar triangular regions and extend the approach to the generation of guaranteed continuous vector fields and planning in multi-terrain environments. We rely on their method and extend it further to work with general triangle meshes that represent curved surfaces embedded in 3D space.

7.2 Triangle Strip Planning

The method of triangle strip planning reconstructs the environment from 3D point cloud data and generates a triangle mesh of the surface. The path is extracted as triangle strip, connecting the start to the goal pose on the mesh. A vector field is computed within each triangle of the triangle strip, such that a smooth path over the mesh is obtained by following the vector field. We first state the problem formulation and then describe the different components of this navigation solution.

7.2.1 Problem Formulation

The navigation solution should generate a feasible and preferably optimal path along a curved surface \mathcal{S} , in accordance with Definition (3.4) and Definition (3.5), by using a triangle mesh as environment representation. The triangle mesh might be given or must be generated from a 3D point cloud.

The triangle strip planner typically acts on an intermediate level in the system architecture. This works under the assumption that there is a human operator or some higher level planner, giving general guidelines on where the robot needs to go. Goal poses requested from a higher level layer can be selected according to an arbitrary coordination policy. In the context of robotic inspection, typical policies set out goal poses for exploration and coverage of the structure.

7.2.2 Environment Representation

3D point clouds provide our basic measurement data for robot localization and mapping, modeling and visualization of the environment. Here our focus is on the representation of the environment; refer to Chapter 2, and to the work of Tâche (2010); Tâche et al. (2011), for a description of a possible robot localization procedure based on scan matching.

Point Cloud Preprocessing

After a 3D point cloud is captured with a LIDAR, the point cloud is filtered and augmented in a preprocessing step prior to the mesh generation. Different filters are applied to improve the point clouds, including equalization of varying point densities, removal of outlier points, and subsampling and trimming in order to reduce the point cloud data. Further methods for smoothing and point cloud simplification, as well as estimation of surface curvature from the point cloud data, are described by Pauly et al. (2002). The tensor voting

framework (see Section 3.4 and Section 8.2.2) can furthermore be applied for augmenting a point cloud. Saliency and structural information are inferred and orientations are estimated.

Augmenting the point cloud by point normals is a common requirement for many mesh generation techniques. Especially implicit surface reconstruction methods rely on robust normal estimation for computing signed distance functions. Besides tensor voting, an alternative method for estimating point normals is the fitting of local planes to the point neighborhoods. A principal component analysis (PCA) on the scatter matrix of the point neighbors estimates the point normal as the eigenvector of the scatter matrix with the smallest corresponding eigenvalue. The resulting normal estimates need to be adjusted to obtain consistent normal orientations among neighboring points. Single point clouds recorded from rotating laser range finders include some inherent additional information about the organization of the point cloud. The points must be visible for the sensor, the point cloud origin coincides with the sensor position, and the recorded data is associated with a particular position of the moving sensor. The point normals can be oriented by using viewpoint information in this case. Alternatively, for multiple point clouds and no available additional sensor information, globally consistent orientations of the normals are obtained by propagating the normal orientations over a minimum spanning tree (Hoppe et al., 1992).

Mesh Reconstruction and Postprocessing

We selected local sensor-centric as well as global, explicit as well as implicit state-of-the-art surface reconstruction methods for our investigation. Some implementations follow the selected method closely, some are implemented with minor modifications.

Irregular Triangular Meshes. The first evaluated surface reconstruction method is the *irregular triangular mesh* generation method (ITM). The method is based on the ideas of Gingras et al. (2010) for mesh generation for robots in rough terrain. ITM is a local explicit approach; it is sensor-centric, i.e., it works on a single laser scan with additional viewpoint information. Points are explicitly connected by triangles through Delaunay triangulation (see also Section 3.3.1). First, the 3D point cloud is transformed into spherical coordinates and a 2D Delaunay triangulation is applied to the two angular coordinates ϕ and θ , omitting the range component. The connectivity among the points is established in this first step. The resulting triangles are close to equilateral as the Delaunay triangulation maximizes the minimal angle of

each triangle. In a second step, range information is added back to generate the final triangle mesh in 3D space. The established connectivity from the first step remains valid. The periodicity in the angle of the spherical coordinates is retained in the generated mesh by duplication and identification of (ϕ, θ) coordinate pairs when the 2D Delaunay triangulation is performed.

ITM interpolates the point cloud, i.e., it keeps the original points of the point cloud as vertices of the mesh and thus exactly represents the real input data. The mesh generation works for non-uniform point densities but is sensitive to noisy data. Typically, the generated meshes contain holes and artifacts, such as incorrect triangle connections at the point cloud boundary or at occluded regions. Meshes obtained by ITM require postprocessing in order to be usable for robot navigation. We implemented a mesh filter after suggestions by Gingras et al. (2010) for the removal of incorrect artifact triangles.

Fast Triangulated Surfaces. The *fast triangulation mesh* generation method (FTM) implements the explicit surface reconstruction method presented by Marton et al. (2009). The method is based on a greedy algorithm that works under the principle of incremental surface growing: a start triangle is created and new triangles are formed and added explicitly until all points in the point cloud are included in triangles in the mesh, or no more valid triangles can be constructed. The triangulation method partially relies on the triangulation algorithm of Gopi and Krishnan (2002). First, the k -nearest neighbors of a single point in the point cloud are selected. In a second step, the points in this neighborhood are projected onto a plane, which is fitted to the points in the neighborhood. Finally, new triangles are built by connecting the points through edges, while visibility and maximum and minimum angle criteria set constraints on the triangle creation. FTM, similar to ITM, is a fast surface reconstruction method that works for non-uniform point densities and provides near real-time performance. It offers improved robustness against noise but still results in much rougher meshes than the implicit methods. The incremental nature of FTM promises the straightforward dynamic extension of a mesh by point clouds from new laser scans.

Meshing with RBF- and MLS-Based Distance Functions. The methods we describe next are global implicit surface reconstruction methods based on differently defined signed distance functions. The *extended marching cubes mesh* generation method (ECM) implements two common choices of signed distance functions: the radial basis function (RBF) (Carr et al., 2001) and moving least-squares (MLS) (Alexa et al., 2001). Once the distance

function computation has completed, the triangle mesh is retrieved by the *marching cubes algorithm*. We use an extended marching cubes implementation (Kobbelt et al., 2001), together with triharmonic or piecewise polynomial RBFs and MLS kernels with Gaussian or Wendland weight functions, in our ECM method.

In contrast to explicit methods, implicit methods generate a triangle mesh as a global approximation of an input point cloud. Whereas RBF-based methods compute an approximation globally, MLS-based methods achieve the approximation by local computation. Different selections of the kernel functions differ in their local or global support, and determine the overall smoothness of the resulting signed distance function and the generated triangle mesh. Implicit methods reconstruct watertight or closed surfaces. In the context of robot navigation, this is advantage as well as disadvantage: noisy regions of the point cloud, regions of highly irregular density or with missed parts can widely be recovered, but the approximation may incorrectly fill in a hole or opening that is truly present in the real surface. Therefore, implicit methods as well require postprocessing of the generated triangle mesh. Mesh filters can use prior knowledge of the surface and additional information extracted from the augmented point cloud or the generated mesh. The mesh could be segmented by fitting characteristic geometric shapes (Schnabel et al., 2007; Vaskevicius et al., 2010) and each of the segmented parts could be checked against the point cloud. Alternatively, one or several regions of connected triangles may be grown and removed from the mesh for areas with missing underlying point cloud data.

Poisson Surface Reconstruction. The *Poisson surface mesh* generation method (PSM) represents a global implicit method, which is based on the Poisson surface reconstruction algorithm introduced by Kazhdan et al. (2006). The Poisson surface reconstruction algorithm is one of the most recent and best performing surface reconstruction algorithms in computer graphics literature to date.

The formulation of PSM relies on the indicator function, a function that distinguishes inside from outside space in a 3D point cloud recorded from a surface of an object. The gradient of such an indicator function can be considered as a vector field that takes non-zero values solely in the proximity of the real surface. The indicator function is found as the function whose gradient approximates the vector field computed from the sample points. By applying the divergence operator, a Poisson equation results, which can be solved efficiently by conventional Poisson solvers (Kazhdan et al., 2006). PSM's basis functions are compactly supported and the implicit function is

inherently constrained at all spatial points without addition of off-surface constraints. Similar to ECM methods, the PSM method depends on augmented point clouds with consistently oriented point normals, is apart from that robust against noisy data, and results in the generation of watertight meshes, including all the associated benefits and drawbacks discussed above.

When a triangle mesh has been generated, the mesh usually needs to be further processed by mesh filters to improve the mesh's navigability prior to path planning. For the postprocessing, we use the QSLim mesh simplification algorithm (Garland and Heckbert, 1997). Other mesh simplification methods could alternatively be used. In the mesh simplification step, the mesh is remeshed and the triangles of the mesh are reasonably scaled for navigation.

7.2.3 Path Planner and Robot Control

Planning Triangle Strip Paths

The input to the path planner consists of the 6D robot pose, the desired goal pose and the previously generated and processed triangle mesh. There are different ways to create a graph or roadmap from the triangle mesh (see also Section 6.2 in Chapter 6); we construct the graph by either using the triangle centroids or the edge centers of adjacent triangles as vertices. The implemented triangle strip planner then uses the classical A* algorithm to connect the triangles to a strip, and hence generate an initial discrete global path from the start to the goal pose. Weights of traversing cost can further be derived from local mesh properties and are added to the graph in order to optimize the geometry of the triangle strip subject to environment and robot characteristics. Robot kinematics and structural constraints could additionally be considered on the discrete level of the triangle strip planning, in a similar way as it is done for point-based path planning in Chapter 8.

Controlling Robots along Smooth Paths

Graph-based planners, as the ones described above, perform well in terms of finding an optimal global path. However, the resulting paths are discrete, and it is desired to have more continuous paths or trajectories and control laws that are directly applicable to the actual robot. We use a vector field controller which generates vector fields within the planned triangle strip to steer a robot toward the goal pose, which is contained in the terminating goal triangle.

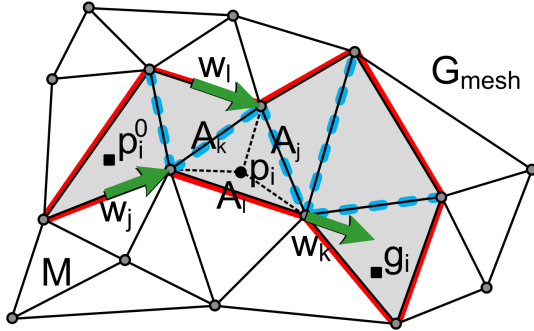


Figure 7.1: Triangle strip path and vector field generation. The combination of vectors \mathbf{w}_j , \mathbf{w}_k and \mathbf{w}_l generates the vector field $\mathbf{u}(\mathbf{p}_i)$ for controlling a robot r_i at position \mathbf{p}_i on the triangle strip from the start at \mathbf{p}_i^0 to the goal at \mathbf{g}_i .

In our case of vector field generation for surfaces $\mathcal{S} \subset \mathbb{R}^3$, we first unwrap the extracted triangle strip and transform it to \mathbb{R}^2 . This unwrapping is inspired by the manifold representation of Howard et al. (2006), has some similarities to the transformation to star-shaped domains of Chapter 5, and corresponds to a transformation of a kinematic chain formed by the triangles of the strip. The m triangles in the strip, with locally defined frames T_i and T'_i , $i \in \{1, \dots, m\}$, are the links of the chain; the edges of two adjacent triangles are the revolute joints. The geometry of the triangles defines the relative translation vectors $\mathbf{t}_{T_i T'_i}$ and rotation matrices $\mathbf{R}_{T_i T'_i}$ for each triangle, and the angles between two adjacent triangle faces give the rotation matrices $\mathbf{R}_{T_{i+1} T'_i}$. The transformed 2D triangle strip can then be further processed using the continuous vector field generation method from Pimenta et al. (2007) and Pereira et al. (2009). The generated vector field is fully continuous in the triangle strip unfolded to the common plane, and only exhibits discontinuities in the relative orientation of adjacent triangle faces in 3D space. This discontinuities are inherent to the piecewise linear approximation of the continuous curved surface by the triangle mesh. Possibilities to achieve continuity in orientation could be found by studying interpolation schemes for interpolating between the orientations of two adjacent triangle faces.

We restate the basics of the vector field generation in the following; for details and a mathematical analysis refer to the work of Pimenta et al. (2007) and Pereira et al. (2009). The vector field is computed by interpolation within each triangle face. For each triangle with triangle vertices v_j , v_k and v_l , a set of three base vectors, \mathbf{w}_j , \mathbf{w}_k and \mathbf{w}_l , is chosen, such that the

vectors are guaranteed to always point along the triangle strip toward the goal without having positive projections to the lateral outward normals of the strip (see Figure 7.1). It is further possible to adjust the single base vectors by additional triangle split operations, such that a fully continuous vector field over the whole triangle sequence is always guaranteed. Given the partial triangle areas A_j , A_k and A_l , we can create the vector field

$$\mathbf{u}(\mathbf{p}_i) = \frac{A_j \mathbf{w}_j + A_k \mathbf{w}_k + A_l \mathbf{w}_l}{A_j + A_k + A_l}, \quad (7.1)$$

for each triangle of the triangle strip from the start to the goal. For a holonomic point robot modeled after Equation (3.16), a gradient descent controller can then be obtained by setting $\dot{\mathbf{p}}_i = \mathbf{u}(\mathbf{p}_i)$.

7.2.4 Extensions

There are several interesting extensions possible with respect to triangle strip planning. Currently, our mesh generator and triangle strip planner are both static. Incremental meshing (Newcombe et al., 2011) and replanning algorithms (Koenig and Likhachev, 2002) provide solutions for more dynamic computations, e.g., to handle dynamically changing environments more efficiently. Another future direction of research is toward obstacle negotiation and avoidance, including 3D collision checking against the triangle mesh. A triangle strip passing a step obstacle could furthermore be aligned with the edges of the step. The vector field could be designed to point into directions perpendicular to the edge tangents, or perpendicular to the edges of the triangles (Lindemann and LaValle, 2005), which would offer a possibility to meet the constraints of a climbing robot for obstacle negotiation (refer to the specifications of the MagneBike robot in Chapter 2, and the inclusion of such constraints under the navigation solution proposed in Chapter 8). The following extensions have been realized so far.

Nonholonomic Vehicles. The case of nonholonomic robots, such as MagneBike, is handled in the triangle strip planner by feedback linearization (Lau-
mond, 1998; Pereira et al., 2009). Defining a reference point different from the robot’s center of mass allows to control a nonholonomic robot similar to a holonomic robot via the vector field of Equation (7.1) as control input.

Covering Triangle Strip Paths. Coverage path planning is included in the navigation solution by an incremental single triangle strip generation method, based on the method of Gopi (2004). Examples of planned covering

triangle strip paths are shown in Figure 6.11 of Chapter 6 on the bottom right.

7.3 Results

In the following, we show representative examples for generated triangle meshes and triangle strip paths. The navigation solution was tested in simulations using synthetic 3D mesh models as well as real 3D point clouds recorded with the Hokuyo URG-04LX and Hokuyo UTM-30LX laser range finders. The navigation solution is implemented in C++ and can also be interfaced from ROS (see Section 2.1.2 for further details on the test environments). The results here presented are qualitative and are intended to demonstrate the navigation solution's general applicability.

7.3.1 Evaluation of Triangle Strip Planning

Synthetic 3D Mesh Models. The triangle strip planner and continuous vector field controller have been tested for mesh models of varying complexity. First, the triangle strips are planned and the vector field is generated along the strip; then the continuous robot trajectories are simulated. Figure 7.2 shows a triangle strip path and a vector field generated on the Stanford Bunny mesh model.

LIDAR Scans of Real Steam Chest. We used 3D point clouds recorded from the steam chest environment and steam chest mock-up (see also Figure 2.1 and Figure 2.4 in Chapter 2), and generated triangle meshes with the different mesh generation methods of Section 7.2.2 above. Some typical characteristics of the evaluated mesh generation methods are shown by the examples of generated triangle meshes in Figure 7.3. ITM may generate a highly irregular mesh, which must be improved by artifact removal and remeshing. ECM leads to a much more uniform mesh but may suffer from spurious surfaces caused by failures in the normal estimation at points of outliers or abrupt density changes. FTM generates the surface accurately but may need substantial postprocessing to render the mesh navigable. PSM robustly generates a watertight mesh. However, the loss of tube openings must be considered in navigation. The point clouds were trimmed and down-sampled to 10'000 points, and normals were estimated where needed but no additional preprocessing was applied. Further preprocessing or tuning of the individual mesh generation methods may improve the results.

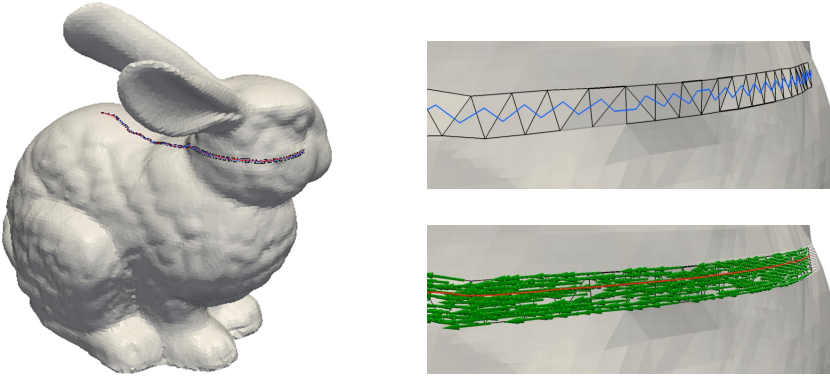


Figure 7.2: Triangle strip path and vector field on Stanford Bunny. Left: Stanford Bunny mesh model and planned global triangle strip path. Right: Extracted triangle strip and discrete path (blue); generated vector field (green) and continuous robot trajectory (red).

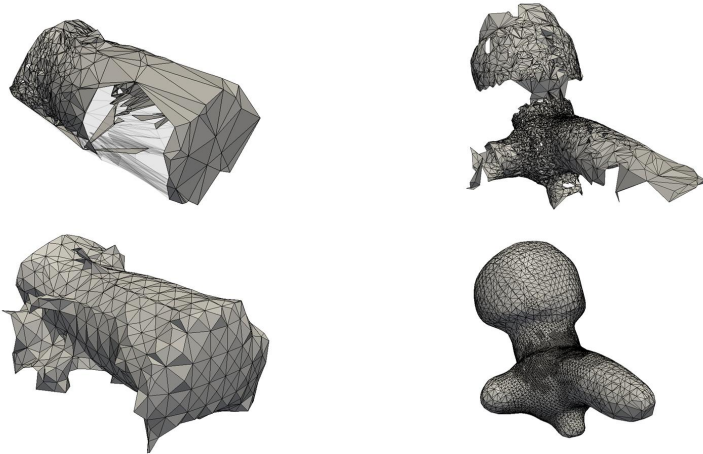


Figure 7.3: Generated triangle meshes of industrial structures. Left: Triangle meshes for a pipe, generated by ITM (top left) and ECM (bottom left). Right: Triangle meshes for the steam chest environment, generated by FTM (top right) and PSM (bottom right). Note the loss of the steam chest's openings in the generated mesh.

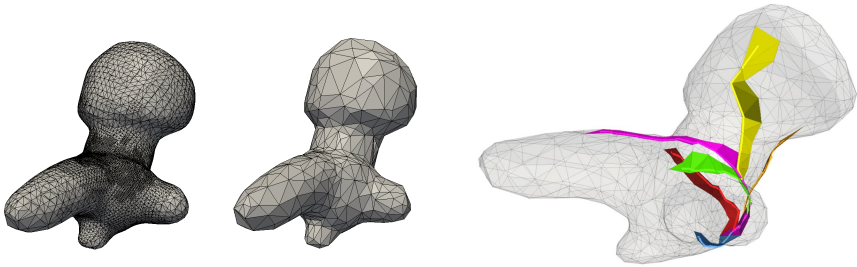


Figure 7.4: Simplified triangle mesh and planned triangle strip paths. Left: Original and simplified triangle meshes for the steam chest. The mesh is reduced from 25'000 to 1'000 faces by QSlim. Right: Six transitioning triangle strips are planned from the original scan location to different goal regions on the structure.

The generated triangle meshes were simplified by QSlim (Garland and Heckbert, 1997). Figure 7.4 on the left shows a triangle mesh before and after simplification; on the right, six transitioning triangle strip paths, all starting at the original robot position, where the underlying point cloud was recorded, and leading to different goal regions on the mesh, are shown. A critical parameter for the navigation solution is the mesh resolution. The optimal resolution depends on the size of the robot as well as the maximum surface curvatures encountered in an environment. Limitations are given for curvatures that are large with respect to a robot's size. The resolution must always be adjusted for a given environment.

Another example of a transitioning triangle strip path and several covering triangle strip paths for the generation of sweeping patterns are shown in Figure 6.11 of Chapter 6.

7.4 Summary

This chapter presents a solution using mesh-based path planning for navigating mobile robots on curved surfaces. Mesh generation methods model the curved surface as triangle mesh and the triangle strip planner plans discrete paths from start to goal regions as a sequence of triangles. Feedback control is directly added to the planning procedure in constructing a continuous vector field. The hybrid approach seems promising as it incorporates several favorable elements for robot navigation on uneven terrain in general

and for robotic inspection in particular, such as mesh representation, efficient discrete planners, and robustness through integrated feedback control.

Several state-of-the-art surface reconstruction methods have been applied to 3D point clouds recorded from industrial tube-like structures, including the steam chest environment. Whereas explicit methods require effective filtering and outlier removal methods in order to create a navigable mesh, implicit methods rely on robust normal estimation and means for detecting missed real features because of too strong approximation and smoothing.

Tasks of an inspection robot are versatile: they span from exploration and mapping to surface coverage and search for defects, to the analysis and repair of defects at specific points in the structure. The underlying problem is always to plan a path on the surface. Depending on which of the tasks needs to be performed, the focus is either on planning a transitioning path to a goal region or on planning a covering path that moves the robot over the complete area. We address both of the problems by planning triangle strip paths over the surface.

Chapter 8

Point-Based Path Planning on Curved Surfaces

In Chapter 7 we introduced a navigation solution for curved surfaces in 3D space which relies on the generation of a triangle mesh from point cloud data and subsequent path planning on the mesh. In this chapter we address the question of how to plan paths more directly on the 3D point cloud without the additional need for an intermediate mesh representation.

Point clouds represent a common input to a mobile robot system. 3D point clouds of several hundred thousand points are recorded by single scans of laser range finders, such as rotating Hokuyo, SICK or Velodyne laser range finders. During the acquisition process, scans are typically aligned and merged to build an overall 3D point cloud representation; methods like Iterative Closest Point (ICP) are used (Besl and McKay, 1992). The point clouds provide samples of the real 3D environment, and as such, they contain noise and distortions as well as sampling irregularities. Generating a mesh representation from such a point cloud raises additional computational expenses. When operating on point clouds which are affected by heavy noise or distortion, further challenges involve the risk of breaking the topology at mesh generation or building in inconsistencies during the following process of mesh management, i.e., while updating an existing mesh with a new partial mesh by alignment and merging.

Path planning is always dependent on the representation of the workspace in which the planning is performed, and thus the quality of the representation has direct influence on the planning performance. Non-degenerate point

clouds¹ can be processed in similar ways to meshes for many operations (e.g., simplification, normal estimation) and must be considered as an alternative for environment representation.

We present two navigation solutions which investigate point-based path planning. Point clouds, unlike mesh-based representations, intrinsically do not provide connectivity information. As connectivity is inherently central to path planning, ways for establishing connectivity throughout the point cloud need to be found.

Both approaches build on a tool originating from computer vision known as *tensor voting* (refer also to Section 3.4 in Chapter 3). Tensor voting infers underlying structure, and estimates saliency and orientation of curves and surfaces. The resulting augmented point clouds form the basis for connecting or grouping points, or for outlier rejection and detection of discontinuities, such as holes or edges, in the surface.

The first navigation solution assumes a dense or densified point cloud. *Dense point cloud planning* establishes connectivity by subdividing the augmented point cloud into a regular grid structure and constructing a graph incrementally. A specialized graph-based planner connects the successive discretized robot poses along the surface into a 6-DoF path, considering kinematic as well as geometric constraints. The 6-DoF path is then transformed from 3D space into 2D space by projecting movements into local tangent planes of the surface, which makes 2D trajectory tracking for robot control possible.

The second navigation solution is designed to work for sparser point clouds. *Sparse point cloud planning* uses surface patches which are fit to the point cloud and provide an analytic continuous description of the local surface. By fitting surface patches, connectivity is established locally within the range of validity of a fit, without the absolute requirement of global consistency. Given a discrete path through the point cloud, the surface patches bridge the gap between subsequent way points. By developing and expanding motion primitives on the surface patches, the robot is finally controlled on the surface.

We have presented the first navigation solution for dense point cloud planning in the International Journal of Robotics Research (Stumm et al., 2012). Section 8.1 describes related work in environment modeling on point cloud data and relevant path planning techniques. Section 8.2 gives a detailed

¹By non-degenerate, we mean point clouds that meet some minimum requirements of being sufficiently dense, e.g., such that they remain connected under a given neighborhood relation, that regions with voids and regions with low sampling density are distinguishable, or that variations through noise are significantly lower than variations introduced by real features.

description of the first navigation solution. The second navigation solution for sparse point cloud planning is outlined in Section 8.3; it is still subject to ongoing research and we will limit our discussion herein to a brief method overview. Section 8.4 summarizes the main results from tests of the first navigation solution in simulations and experiments. The chapter is concluded by the summary in Section 8.5.

8.1 Related Work

In computer graphics, the graphics pipeline has been reconsidered in the past decade and points have been rediscovered as powerful graphics primitive. Graphics concepts of modeling, processing and rendering were transferred from polygonal meshes to points, leading to what is called *point-based graphics* (Gross and Pfister, 2007). Pauly et al. (2002), for example, demonstrates simplification methods for point-sampled surfaces, which prove equal in performance to more conventional mesh-based methods. Meshes are supported by current graphics hardware and model the connectivity and topology of a surface explicitly. In contrast, point-based representations reduce the complexity that arises from maintaining this connectivity, and are more robust against noise, sampling non-uniformity or sampling incompleteness. We have taken inspiration from point-based graphics in our undertaking of studying point-based planning.

Smith et al. (2011) proposes an alternative point-based representation, which predicts the underlying surface by Gaussian process regression. This probabilistic formulation uses beam-space parametrization to achieve a non-functional representation; modeling of arbitrary 3D surfaces and adaptive compression of the point cloud become possible. The method demonstrates how point clouds can be sampled actively and reduced upfront by removing redundant data, thereby speeding up computation and lowering memory requirements. Selecting relevant points from the point cloud can also be realized by the fundamentally different approach of tensor voting. Tensor voting is a local and data-driven method, which is based on the Gestalt principles for perceptual organization (Medioni et al., 2000; Mordohai and Medioni, 2006). By exploiting proximity and continuation information, structural saliency and orientation discontinuity in a point cloud are inferred, and points are organized into coherent groups. King (2008) extends tensor voting for the specific application to 3D point clouds and the modeling of 3D environments.

A common approach in robotics is to represent an environment in a regular grid structure (Moravec, 1996). However, memory limitations restrict full grid structures in size and resolution, which becomes more pronounced for voxel grids in three dimensions. Recent approaches use tree-based rep-

representations, such as octrees (Wurm et al., 2010), and offer increased compactness and flexibility regarding multi-resolution and dynamic extensions of the representation. The advantage of regular grid structures is that they directly allow for graph search planning techniques like Dijkstra’s algorithm and heuristic and incremental extensions thereof (see also Section 3.3.2 and Chapter 7). When surfaces in 3D space are modeled with grids, it is particularly desirable that planned paths follow the surface closely. This can be achieved by relaxing the constraint of being restricted to the grid’s vertices and edges (Carsten et al., 2006; Nash et al., 2010). As valid paths must lie on the surface, it is sufficient to consider a narrow offset band in the ambient space around the surface for path generation. The offset band contains the subset of points from the point cloud that tightly surrounds the underlying surface, and can again be modeled by a regular grid—this time defined within the offset band only. Mémoli and Sapiro (2001, 2005) approximates distance functions and geodesics on point clouds with bounded error by using a fast marching algorithm within an offset band.

An alternative way of establishing connectivity over the point cloud is to build a neighborhood graph, connecting neighboring points of the point cloud directly. Popular methods from manifold learning, for instance, build up connectivity and compute shortest paths between data points on the manifold for following dimensionality reduction, e.g., by multidimensional scaling (Tenenbaum et al., 2000). Likewise, neighborhood graphs can be constructed in incremental fashion. Acting on the point cloud, sampling-based planners seem to be a perfect tool to this task. Rapidly-exploring random trees (RRT) (LaValle and Kuffner Jr., 2001) and their asymptotically optimal variant RRT* (Karaman and Frazzoli, 2011) grow a tree from a start to a goal point through space by repeated point sampling and tree extension steps. Extensions to RRT exist that control the Voronoi bias and adapt RRT’s sampling domain (Yershova and LaValle, 2009), which means sampling within a limited space around the surface in our context. In particular, various RRT-based algorithms, such as the constrained bidirectional RRT (Berenson et al., 2009) and AtlasRRT or AtlasRRT* (Jaillet and Porta, 2012), have been proposed in the field of manipulation planning.

All of the above planning techniques approximate geodesic shortest paths. Another important aspect in path planning for mobile robots is the consideration of the vehicle kinematics and nonholonomic constraints. Sampling-based planners like RRTs offer the possibility to incorporate such constraints via steering methods or motion primitives in the tree extension step (Frazzoli et al., 2005; LaValle, 2006). Steering methods and motion primitives, by construction, guarantee motions that are feasible for the given system.

Continuing with the point cloud as our main representation, fitting of geometric primitives offers an option to provide local surface representations that support the generation of motion primitives. Schnabel et al. (2007) extracts basic geometric shapes from a point cloud and reassembles the underlying surface from a predefined set of shape primitives, including planes, spheres, cylinders, cones and tori. Thrun et al. (2004a) recovers multiple flat surfaces from a point cloud solely by fitting planar shape primitives. Modeling with shape primitives results in compact abstractions that are mainly used for the visualization of man-made environments. However, even man-made structures are not entirely composed of these primitives either. Inaccuracies in the surface approximation due to local deviations bear the risk of planning failure when relying on such shape primitives for moving on the surface. The relevance of the chosen set of shape primitives and the committed approximation errors both decrease as the shape primitives become more local and general in scope. For instance, shape primitives that apply more generally result from local fitting of polynomials (Cazals and Pouget, 2005) or quadric surfaces (Vona and Kanoulas, 2011).

8.2 Dense Point Cloud Planning

We first formulate the problem before delving deeper into the individual building blocks of the first navigation solution.

8.2.1 Problem Formulation

Our main objective for point-based path planning is to generate feasible and preferably optimal paths along a curved surface \mathcal{S} by using a 3D point cloud as environment representation. The first navigation solution primarily focuses on the local scale²; the dense point cloud planner must connect the current robot pose via a 6-DoF path to a nearby goal pose on the robot’s visible surrounding surface, which is represented by point samples.

As defined by Definition 3.4, feasibility implies that the resultant paths remain feasible with respect to environment geometry and vehicle kinematics. The paths must be constrained to the perceived surface, and unknown or uncertain areas of low saliency must be avoided. In addition, limits on the path curvatures are to be considered, such that planned paths are sufficiently

²As the experiments of Section 8.4.2 will demonstrate, the first navigation solution is also capable of generating longer and more complex 6-DoF paths throughout large and noisy point clouds at a global scale—thus acting, in fact, as a more global planner, although not primarily designed for this purpose.

smooth to be tracked by a nonholonomic vehicle; the vehicle is modeled according to the state transition equation of Equation (3.14) and obeys the control inputs $\mathbf{u}(t)$. Optimality additionally aims at minimum travel time and increased safety, in minimizing the path length and overall risk of a path. Optimality can be included according to Definition 3.5 by an appropriate cost function.

With respect to the inspection scenario and the MagneBike robots, this means avoiding unknown areas as well as reducing the attack angle when traversing high curvature regions. Sharp bends, including the negotiation of step transitions, are only safe if the curvature lies in the robot's x - z sagittal plane, thus maintaining proper contact between the wheels and the surface (refer to Chapter 2 for specifications of the environments and the robots). These criteria must be satisfied by the path planning algorithm in order to achieve practical results.

8.2.2 Environment Representation

Raw 3D point clouds generated from laser range finders provide sparse un-oriented input tokens, whereas for our application dense oriented data is required for navigation and control.

A point cloud is filtered initially by the same point cloud filters as applied in Chapter 7 for mesh-based path planning. The subsampled and trimmed point cloud is then further processed by tensor voting (see also Section 3.4). Rounds of token reduction and sparse voting produce an augmented point cloud, which serves as a starting point for the navigation solution.

Dense point cloud planning next performs tensor splitting and another round of dense voting in order to densify the point cloud. Tensor voting achieves this densification through voting performed by each input token at every location in a predefined grid structure, essentially providing information at any desired resolution. In addition, the orientation of any directional feature is estimated by tensor voting simultaneously.

Densification and splitting the tensors one more time into their stick, plate and ball components generate dense and regular structural information. The resulting tensors are stored as 3×3 matrices in a 3D voxel grid, which leads to so-called *environmental feature maps*, including surface saliency, surface normal, edge saliency, edge tangent and binary edge maps. The feature maps represent the final augmented point cloud.

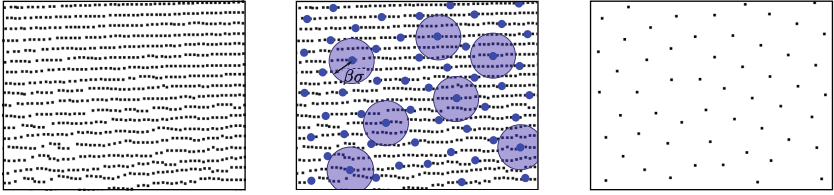


Figure 8.1: Token reduction of LIDAR point cloud. Left: Anisotropic input point cloud. Center: Selected tokens (blue) and a few examples of the subsampling disks, with their radius defined by the masking parameter β and the voting scale parameter σ . Right: Uniform reduced output point cloud.

Token Reduction of LIDAR Point Clouds

As introduced in Section 3.4, token reduction can be applied after the first round of sparse voting has been performed. The point cloud is filtered and reduced, which leads to computational savings in the further process of tensor voting. Thereby it is important not to throw out a lot of pertinent information, compromising the feature extraction. Under the assumption that relevant features, i.e., the strongest votes, occur on the scale of the voting scale parameter σ , the subsampling of tokens should be related to σ as well. Once tokens become too sparse, and the inter-token distance exceeds σ , the saliencies resulting from the tensor voting should reflect a low confidence in these areas. In contrast, in any dense regions where the inter-token distance drops below σ the resulting saliencies should be considerably higher but less dependent on the local density.

Our approach iteratively selects tokens based on highest saliency after sparse voting, and then removes any nearby tokens within a given radius, taken as a fraction of the voting scale, $\beta\sigma$, where the masking parameter β is a positive fraction in the interval $[0, 1]$. As a result, in high density regions, the remaining tokens are spaced such that the inter-token distance is roughly equal to this radius, whereas in low-density regions, the inter-token distance remains the same as before (see Figure 8.1).

Structure Inference for Path Planning

The tensors that result from the final round of dense voting can be decomposed again using Equation (3.13b), which produces dense structural information for each underlying dimension. The final decomposed tensors are defined at every cell over a 3D grid of chosen resolution; they are used to cre-

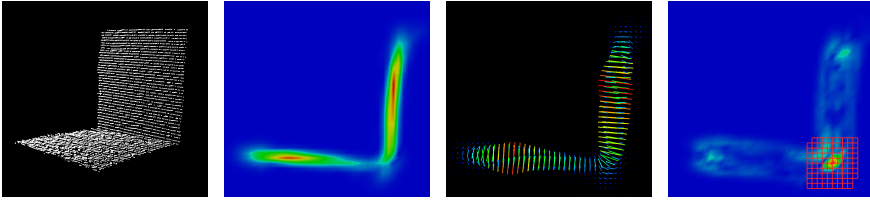


Figure 8.2: Environmental feature maps. Saliency maps and edge maps are extracted from a 3D augmented point cloud for modeling the environment. The augmented point cloud results from tensor voting, including rounds of sparse and dense voting. Left to right: Input tokens of the point cloud, representing an L-shaped metal structure; the original 3D point cloud was recorded with a rotating laser range finder. Cross section of the surface saliency map. Cross section of the surface normal map, scaled by saliency. Cross section of the edge saliency map with an overlay of the binary edge map.

ate a set of environmental feature maps, which serve the dense point cloud planning algorithm for cost computation and navigation.

During tensor decomposition, each structure type has an associated saliency and directionality. For our application of planning on curved surfaces, the environment is expected to consist of only surfaces and edges, therefore limiting our analysis to the final stick and plate tensors. The surface saliency and surface normal maps store surface saliencies and surface normals, which are given by the largest magnitude eigenvalue and eigenvector of the stick tensors, obtained from their 3×3 matrix representations.

Edge information, on the other hand, is first thresholded, and only classified as an edge if the plate saliency is high enough, above a preset edge threshold λ_{edge} . This edge information is stored in an edge saliency map. Any of the map's cells will be treated as a form of obstacle, and so the saliency value is no longer important to us with respect to path planning. A binary edge map is constructed, which distinguishes each grid cell to be either an edge or not. The edge tangent, given by the lowest magnitude eigenvalue and eigenvector of the plate tensors, is relevant for the edge traversals of the robots, and is thus stored in an associated edge tangent map.

The robots are treated as point robots for path planning. In order to avoid collisions and guarantee safe edge traversal, edge obstacles are inflated by a radius, which is chosen to suit the respective robot specifications. Edge inflation is implemented by labeling all the neighboring grid cells of an edge cell in the binary edge map as edge cells as well, i.e., setting their binary

values to “true”. The corresponding tangents are further given by averaging all the tangents from the nearby grid cells that contain the original edge. Figure 8.2 shows an example of the environmental feature maps produced from a real 3D point cloud of an L-shaped structure.

8.2.3 Path Planner

The dense voting in the tensor voting process creates a 3D grid-based environment representation, and path planning utilizes this grid structure in connecting a series of cells throughout the grid to the goal. The connections define a graph $G = \{V, E\}$ over the grid, with nodes $v_i \in V$ representing the grid cells and edges $e_{ij} \in E$ representing feasible transitions to a neighboring node v_j . Each node v_i has an associated node state $\mathbf{x}_i \in \mathcal{X}$, with state space \mathcal{X} , which contains information about the position of a cell in the 3D grid, given by its coordinates (x, y, z) with respect to the world frame W . The best discrete path toward the goal pose can be found by graph search. We use a specialized graph-based path planner which takes constraints into consideration and uses the A* algorithm for search on the graph G , thereby inheriting desirable properties of optimality and resolution completeness guarantees.

The definition of the graph structure G is not straightforward under consideration of constraints. When establishing connectivity between nodes, edge transitions and edge transition costs must be chosen in such a way as to satisfy constraints imposed by the environment geometry and vehicle kinematics. In the following, further details of the graph construction are discussed.

Graph Connectivity

Each grid cell has associated structural information, and now the question arises of how this information can be used to connect the nodes locally in order to build a graph structure, while supporting kinematically feasible movements between cells.

Our specialized graph-based planner performs a lazy grid search on a 6D state space and searches for edge transitions to neighboring nodes with the help of a *discrete control set* with predefined movement vectors \mathbf{m}_B , given in the robot’s local body frame B . These movement vectors account for structural as well as vehicle constraints, as they restrict the path segments to directions that have high probability of being tracked successfully by the robot. The planner exhibits notable parallels to constrained path planning in state lattices, where states are arranged regularly and connected by feasible motion primitives (Pivtoraiko and Kelly, 2005).

Robot mobility is directly linked to the local surface properties, therefore our path planning algorithm must be cognizant of both the robot’s and the structure’s orientations. To achieve this, besides the position of a cell, the node state \mathbf{x}_i must also include orientation information; this leads to a 6D state space, $\mathcal{X} \subset \mathbb{R}^6$. In our discretized state space, orientation is defined as a discretized heading vector $\hat{\mathbf{h}}_W$ with respect to the global frame W . Thus each cell in the 3D grid is actually represented by a set of m_{dir} nodes, where each node v_i has equal position but varying orientation in the state vector \mathbf{x}_i , which correspond to the set of m_{dir} discretized heading vectors at a certain cell position. The node states are illustrated in Figure 8.3; note that for ease of illustration only $m_{\text{dir}} = 6$ different heading vectors are shown here, as opposed to the $m_{\text{dir}} = 98$ in our actual implementation.

The fact that robot motion is constrained to a surface $\mathcal{S} \subset \mathbb{R}^3$, which is a 2D manifold embedded in the 3D workspace $\mathcal{W} \subset \mathbb{R}^3$, reduces the number of nodes to be searched, and only the cells in a narrow offset band in close proximity of the estimated surface need to be considered. Using this, node connectivity is established *on the fly* by restricting transitions to those neighboring nodes, i.e., cell locations and orientations, deemed physically reachable (see Figure 8.3 for further illustration of the concept). This lazy evaluation allows for the direct inclusion of some kinematic constraints while limiting the search space to tangential movements along the estimated surface.

The connections between nodes are established as the nodes are expanded. The connectivity for each node is dependent on the environmental feature maps provided by tensor voting and the discrete control set, along with the robot orientation in the world frame W . There are essentially two different scenarios—either the current node belongs to an edge structure or it does not. The case is determined using the binary edge map, and the planner specifies which of the two control sets has to be applied in each case. Figure 8.4 shows the two discrete control sets and their movement vectors for both scenarios. The selection from among variable control sets is further related to on-demand and graduated fidelity concepts of planning in state lattices (Pivtoraiko and Kelly, 2008).

The transformation between the body frame B and world frame W is deduced using the robot’s heading vector $\hat{\mathbf{h}}_W$ and the local surface normal $\hat{\mathbf{n}}_W$. Once this transformation has been determined, the appropriate set of movements $\mathcal{P}_{\text{move}}$, represented by m_{move} movement vectors $\mathbf{m}_{k,B}$, $k \in \{1, \dots, m_{\text{move}}\}$, is transformed into the world frame W . Each movement \mathbf{m}_W is discretized by snapping the vector to the position of the closest grid cell whose associated node v_j is a direct neighbor of the current node according to the set of heading vectors. This neighbor node v_j represents a possible child

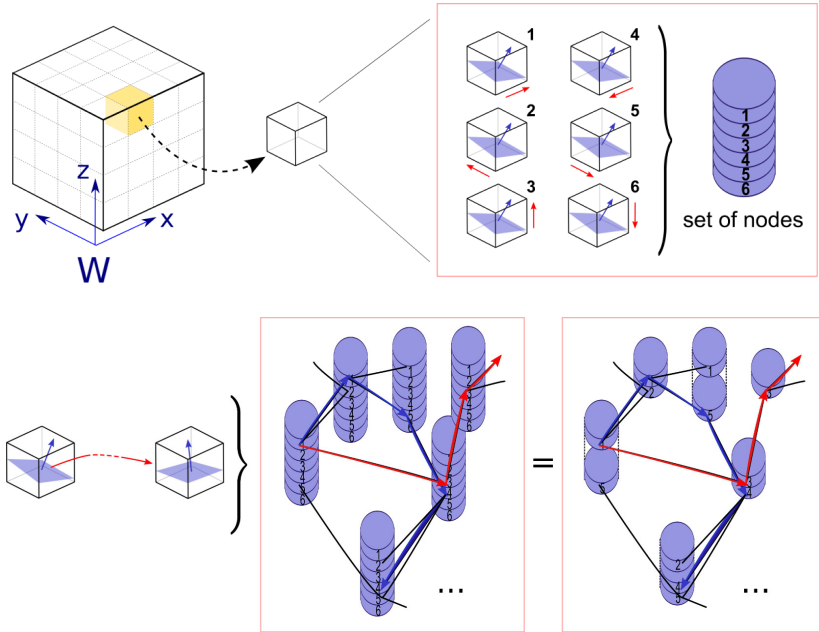


Figure 8.3: Graph connectivity. Top row: The node state \mathbf{x}_i contains the 3D location as well as the potential robot heading vector $\hat{\mathbf{h}}_W$ (here discretized into $m_{\text{dir}} = 6$ directions, shown by vectors drawn next to the cells). Each location has a set of associated nodes, one for each (discretized) direction. Establishing movement from one node (or cell) to another is aided by the structural information given by the tensor voting framework, e.g., by local surface normal estimates. Bottom row: Single nodes from each set of nodes are connected in a graph, taking given constraints in the robot’s and surface’s orientations into account. According to these constraints, paths may go through the same physical 3D location, i.e., the same set of nodes, but with a different heading, and thus remain unconnected, i.e., at different nodes.

node, and the projected movement vector connecting to this candidate child node is given by \mathbf{m}'_W . \mathbf{m}'_W is an element from the set of all possible heading vectors, as illustrated in Figure 8.3. The full node state \mathbf{x}_j , i.e., position and orientation, of the candidate child node v_j is determined accordingly.

Next the surface saliencies are compared to a threshold λ_{surf} . Connections to candidate child nodes with low surface saliency are rejected, and only connections to nodes with strong surface values are permitted. Then the

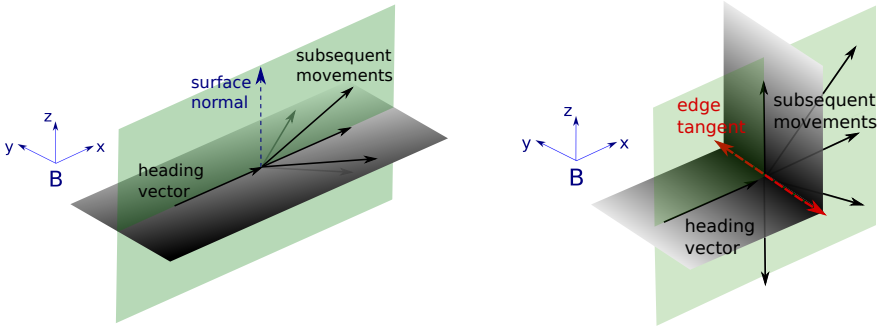


Figure 8.4: Discrete control set. A set of possible movement vectors $\mathcal{P}_{\text{move}} = \{\mathbf{m}_{k,B}\}_{k=1}^{m_{\text{move}}}$, with respect to the robot’s local body frame B , is given for two possible cases. Left: Regular moves on the surface, where the current node is not nearby an edge. Right: Special moves over an edge, where the current node is near an edge of the environment.

turn angle θ is calculated by projecting the movements into the local surface plane (see also Equation (8.3) further below), and the graph connectivity is only established for turns smaller than a threshold λ_{turn} . Finally, the cost of this connection is evaluated to determine whether the node will be added to the priority queue for expansion. The complete procedure for graph connectivity construction is summarized by Algorithm 10.

The realized graph connectivity ensures perpendicular traversal of edges in the environment, limited turn angles, and staying within the close neighborhood of the best surface estimate. Once a valid connection between two nodes is established, costs based on the risk of traversing a connection are assigned to the graph edge e_{ij} in order to meet the remaining requirements of path planning.

Cost Functions and Heuristics

After a node’s children have been found using the process described above and the node has been expanded and removed from the priority queue, the edge transition cost of traveling to each child node needs to be evaluated in order to determine if this child node should eventually be added to the priority queue. The choices of cost functions for the edge transition cost vary by applications.

Algorithm 10 Graph Connectivity Construction

Require: Graph G , to be updated by establishing new connectivities e_{ij} between the current node v_i and potential neighboring nodes v_j . Node v_i has node state \mathbf{x}_i with position $\mathbf{p}_i \in \mathbb{R}^3$ and orientation $\widehat{\mathbf{h}}_W$, given with respect to the world frame W^1 . Further inputs are the predefined sets of movements $\mathcal{P}_{\text{move}}$, and estimates of surface normals $\widehat{\mathbf{n}}_W$ and edge tangents $\widehat{\mathbf{t}}_{\text{edge},W}$ within a neighborhood of v_i , e.g., as a result of tensor voting.

```

1: if movement == SURFACE then
2:    $\widehat{\mathbf{b}}_{z,W} \leftarrow$  current surface normal  $\widehat{\mathbf{n}}_W$ 
3:    $\widehat{\mathbf{b}}_{x,W} \leftarrow$  current heading vector  $\widehat{\mathbf{h}}_W$ 
4:    $\widehat{\mathbf{b}}_{y,W} \leftarrow \widehat{\mathbf{b}}_{z,W} \times \widehat{\mathbf{b}}_{x,W}$ 
5:   Set of moves  $\mathcal{P}_{\text{move}} \leftarrow$  regular moves on surface (Figure 8.4, left)
6: else if movement == EDGE then
7:    $\widehat{\mathbf{b}}_{z,W} \leftarrow$  current surface normal  $\widehat{\mathbf{n}}_W$ 
8:    $\widehat{\mathbf{b}}_{y,W} \leftarrow$  current edge tangent  $\widehat{\mathbf{t}}_{\text{edge},W}$ 
9:    $\widehat{\mathbf{b}}_{x,W} \leftarrow \widehat{\mathbf{b}}_{y,W} \times \widehat{\mathbf{b}}_{z,W}$ 
10:  Set of moves  $\mathcal{P}_{\text{move}} \leftarrow$  special moves over edge (Figure 8.4, right)
11: end if
12: Compose rotation matrix  $\mathbf{R}_{WB} = \begin{bmatrix} \widehat{\mathbf{b}}_{x,W} & \widehat{\mathbf{b}}_{y,W} & \widehat{\mathbf{b}}_{z,W} \end{bmatrix}$ 
13: for all possible moves  $\mathbf{m}_B \in \mathcal{P}_{\text{move}}$  do
14:   Transform movement into world frame,  $\mathbf{m}_W = \mathbf{R}_{WB} \mathbf{m}_B$ 
15:    $\mathbf{m}'_W \leftarrow$   $\mathbf{m}_W$  “snapped” to the closest grid cell by projection
16:   Possible child node  $v_j$  at position  $\mathbf{p}_j \leftarrow \mathbf{p}_i + \mathbf{m}'_W$ 
17:   if surface saliency  $> \lambda_{\text{surf}}$  and turn angle  $< \lambda_{\text{turn}}$  then
18:     Establish edge connectivity  $e_{ij}$ 
19:     Expand current node by evaluating the cost of transition  $\mathbf{m}'_W$ 
20:   end if
21: end for

```

¹ The robot’s body frame B is defined by the unit vectors along the coordinate axes $\widehat{\mathbf{b}}_{x,W}$, $\widehat{\mathbf{b}}_{y,W}$ and $\widehat{\mathbf{b}}_{z,W}$, represented in coordinates of the world frame W .

For the inspection scenario and the MagneBike robots we determined the following four contributions to the edge transition cost between two nodes:

- cost function w_{dist} , rating the distance (or step size)
- cost function w_{surf} , rating the surface saliency
- cost function w_{turn} , rating the turn angle
- cost function w_{curv} , rating the relative surface curvature

The overall edge transition cost is then given by the total cost function

$$w_{\text{total}} = k_1 w_{\text{dist}} + k_2 w_{\text{surf}} w_{\text{dist}} + k_3 \frac{w_{\text{turn}}}{w_{\text{dist}}} + k_4 w_{\text{curv}}, \quad (8.1)$$

where each individual cost function is weighted appropriately by the scalars k_1 , k_2 , k_3 and k_4 .

The rationale behind each contribution to the total cost function is described next, with the exception of the distance cost, which is simply given by the Euclidean distance between the position of a node and its child node, approximating the edge length by the distance along a straight line. The heuristic for the cost-to-go in the A* algorithm is similarly given by the Euclidean distance from the position of a current node to the position of the goal node in the 3D workspace \mathcal{W} .

Surface Saliency. In practice, when using a LIDAR point cloud, the surface saliencies $s(\mathbf{p}_i) \in \mathbb{R}$ at positions \mathbf{p}_i of node v_i vary substantially throughout the point cloud and saliencies with values above a relatively low threshold typically correspond to positions \mathbf{p}_i close to the actual surface \mathcal{S} . High saliency regions usually result from dense sampling in these areas. The token reduction strategy described in Section 8.2.2 helps to reduce this saliency variation but does not eliminate it completely. Higher saliency regions should have some preference, however not so much that the path meanders unnecessarily in order to traverse densely sampled areas. Therefore the relative surface saliency is raised to an exponent (tuned empirically to $n = 4$ in our experiments), penalizing low saliency regions much more than high saliency regions, which leads to the cost function

$$w_{\text{surf}} = \left(\frac{s_{\text{max}} - s(\mathbf{p}_i)}{s_{\text{max}}} \right)^n, \quad (8.2)$$

with $n \in \mathbb{N}_{>0}$, and maximum surface saliency given by the maximum value s_{max} found in the grid structure. Note that the surface cost is multiplied by the cost of the step size in the total cost calculations, since longer steps over low confidence areas should have a higher penalty than shorter steps.

Turn Angle. Turn angles are calculated by projecting the movement vector \mathbf{m}_W and heading vector $\hat{\mathbf{h}}_W$ into a local surface plane, leading to $\mathbf{m}_W^{\text{avg}} = \mathbf{m}_W - \hat{\mathbf{n}}_W^{\text{avg}} (\mathbf{m}_W \hat{\mathbf{n}}_W^{\text{avg}})$ and $\mathbf{h}_W^{\text{avg}} = \hat{\mathbf{h}}_W - \hat{\mathbf{n}}_W^{\text{avg}} (\hat{\mathbf{h}}_W \hat{\mathbf{n}}_W^{\text{avg}})$. The local surface plane is given by the normal vector $\hat{\mathbf{n}}_W^{\text{avg}}$, which results from averaging the surface normals located at the parent node $\hat{\mathbf{n}}_{i,W}$ and its child nodes $\hat{\mathbf{n}}_{j,W}$ respectively, which gives $\mathbf{n}_W^{\text{avg}} = (\hat{\mathbf{n}}_{i,W} + \hat{\mathbf{n}}_{j,W})/2$, with $i \neq j$, $i, j \in \mathbb{N}$. This local surface plane can be thought of as the tangent plane $T_{\mathbf{p}_{ij}^{\text{avg}}} \mathcal{S}$, with $\mathbf{p}_{ij}^{\text{avg}}$ representing an intermediate position on the surface \mathcal{S} where the surface normal just assumes $\hat{\mathbf{n}}_W^{\text{avg}}$. The cost function is then obtained by taking the absolute value of the angle between the two projected vectors

$$\Delta\theta = \cos^{-1} \left(\frac{\mathbf{m}_W^{\text{avg}} \cdot \mathbf{h}_W^{\text{avg}}}{|\mathbf{m}_W^{\text{avg}}| |\mathbf{h}_W^{\text{avg}}|} \right) = \cos^{-1} \left(\hat{\mathbf{m}}_W^{\text{avg}} \cdot \hat{\mathbf{h}}_W^{\text{avg}} \right), \quad (8.3)$$

$$w_{\text{turn}} = |\Delta\theta|^2.$$

Similar to the surface cost calculation, the resulting angle is squared, such that two smaller turns are preferred over one larger turn. In addition, the turn cost is divided by the step size in the total cost calculations, in order to prefer more gradual turns.

Relative Surface Curvature. The MagneBike robots are capable of traversing curved surfaces, however, areas of high curvature can be problematic if approached at the wrong angle. Therefore the angle between the plane of curvature and the x - z plane of the robot's local body frame B should be minimized in areas of high curvature (see Figure 8.6, left). The curvature cost results from the product of the magnitude of change in the surface normal, $\Delta\hat{\mathbf{n}}_W$, and the approach angle α , and is given by the cost function

$$w_{\text{curv}} = |\Delta\hat{\mathbf{n}}_W| |\alpha|$$

$$= \cos^{-1} (\hat{\mathbf{n}}_{i,W} \hat{\mathbf{n}}_{j,W}) \cos^{-1} \left(\frac{\hat{\mathbf{n}}_{i,W} \times \hat{\mathbf{n}}_{j,W}}{|\hat{\mathbf{n}}_{i,W} \times \hat{\mathbf{n}}_{j,W}|} \cdot \hat{\mathbf{b}}_{y,W} \right), \quad (8.4)$$

where $\hat{\mathbf{n}}_{i,W}$ and $\hat{\mathbf{n}}_{j,W}$ denote again the surface normals at the positions of the parent and child nodes, and $\hat{\mathbf{b}}_{y,W}$ is the y -axis of the robot's local body frame, represented in the coordinates of the world frame. Intuitively, the change in surface normal is inversely weighted by the step size, however the overall curvature cost is directly weighted by the step size, therefore eventually canceling out any dependence on the step size in the total cost calculations.

8.2.4 Robot Control

Once a feasible 6-DoF path on the surface has been generated by the path planner, a control strategy is required to steer the robot from the start to the goal pose. Path following or trajectory tracking on curved surfaces, and in constrained environments in general, is no simple task. Next we describe a method that transforms way points of the 6-DoF path, which are given by a sequence of connected nodes, into \mathbb{R}^2 with respect to a fixed 2D control frame. The basic idea here is that the robot can be modeled in two dimensions, which allows for the use of standard and more widely studied 2D control schemes. A trajectory tracking controller based on nonlinear feedback design is then used together with a front-wheel-drive bicycle model to demonstrate how the computed trajectories can be tracked in the case of the MagneBike robots. Note the similarity to Chapter 7 where the triangle strip path is flattened by transforming it to the plane in order to generate the controlling vector field.

Path Dimensionality Reduction

Although the robot motion is in 3D space, the robot is always constrained to travel on a 2D manifold. Therefore, similar to the idea by Furgale and Barfoot (2010), the path generated by the A* algorithm can be transformed into a lower dimensional space before a control strategy is applied. This is done by iteratively projecting the two movement vectors $\mathbf{m}'_{i,W}$ and $\mathbf{m}'_{j,W}$ of a parent and its child node at positions \mathbf{p}_i and \mathbf{p}_j in the grid onto their local surface plane, i.e., onto the associated tangent plane $T_{\mathbf{p}_{ij}}^{\text{avg}}\mathcal{S}$. The tangent plane is found by averaging the surface normals of the corresponding parent and child nodes as described under Section 8.2.3 above. The relative turn angle $\Delta\theta$ between the two movement vectors is calculated according to Equation (8.3). The 2D coordinates of the i th-way point in the path sequence, (x_i, y_i) , are then iteratively given by the 2D coordinates (x_{i-1}, y_{i-1}) of the previous way point and the projected 2D movement vector \mathbf{m}_i , rotated by $\Delta\theta_i$ with respect to the previous movement vector. This formulation is shown in Figure 8.5 on the left, and further described by

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + |\mathbf{m}_i| \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix}, \quad (8.5)$$

$$\theta_k = \theta_{i-1} + \Delta\theta_i,$$

with $i \in \mathbb{N}_{>0}$. At start, the coordinates (x_0, y_0) of the path sequence are initialized at the origin of the 2D control frame and θ_0 is initialized to zero.

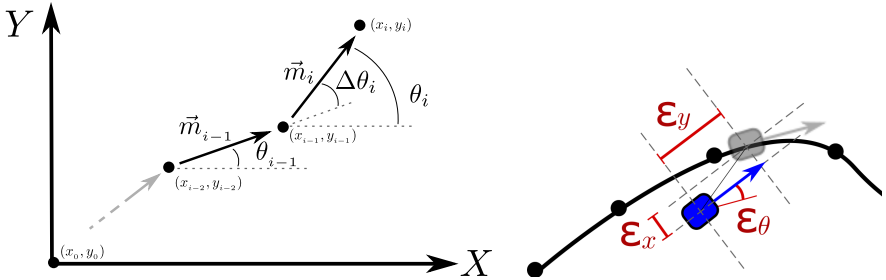


Figure 8.5: Robot control. Left: Iterative computation of 2D way points based on the projected 3D movement vectors and the corresponding turn angles. Right: Tracking a virtual reference bike through nonlinear feedback of the pose error.

The alignment of the 2D movements based on the relative yaw angles between them essentially leads to *unwrapping* the path from the surface. An illustration of this transformation can be seen in Figure 8.6 on the right.

Now that the trajectory has been mapped into 2D space, the system model can be treated in two dimensions as well. The MagneBike robot is approximated by a simple front-wheel-drive bicycle, with control inputs given by the speed $v_{\text{front}}(t)$ and steering angle $\phi(t)$ of the front wheel. The kinematic models for bicycle and car-like robots were introduced in Section 3.5.1 of Chapter 3. One possible method of control, using state feedback to track a parametrized trajectory, is outlined in the following.

Trajectory Tracking

In order to control the robot along the given trajectory, a set of nonlinear feedback controls for trajectory tracking are applied. This is realized for MagneBike by using the kinematic model in Equation (3.17) with control inputs given by Equation (3.20b), and following a virtual reference bike with known state along the path as suggested by Samson and Ait-Abderrahim (1991). To establish the state of the reference bike for all time, a B-spline is fitted through the 2D way points, creating a continuous parametrized path from which the state of the virtual reference bike can be inferred. For simplicity, the reference bike will be assumed to follow the spline at a chosen constant velocity v_{ref} . Once the state of the reference bike has been determined, a nonlinear feedback law can be used to stabilize the position and orientation error of the MagneBike robot, relative to the reference bike, to zero. Position and orientation feedback are assumed to be provided by the robot's

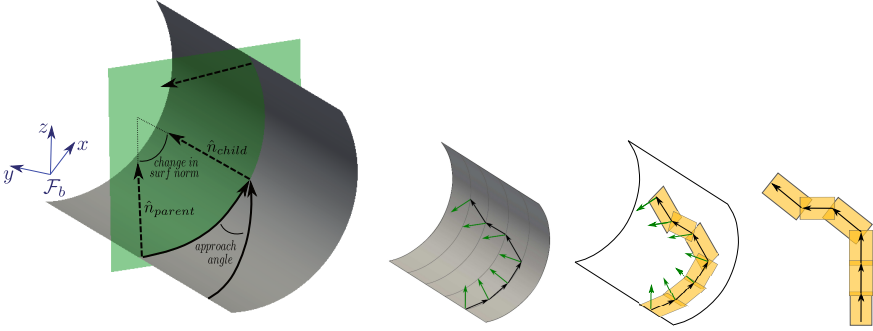


Figure 8.6: Paths on curved surfaces. Left: Traveling at large angles with respect to the curvature plane is penalized with higher costs. Right: Transforming a 6-DoF path on a curved surface from \mathbb{R}^3 to \mathbb{R}^2 by projecting each movement vector \mathbf{m}'_W into its associated tangent plane $T_{\mathbf{p}^{avg}} \mathcal{S}$.

localization system (see also Chapter 2). The state vector of the system for trajectory tracking is now given by $\mathbf{x}_{\text{track}} = [\epsilon_x \ \epsilon_y \ \epsilon_\theta]^T$, where ϵ_x and ϵ_y represent the relative position of the reference bike in MagneBike's local frame B , and $\epsilon_\theta = \theta - \theta_{\text{ref}}$, where θ and θ_{ref} denote the respective orientations of the bike and reference bike in the 2D control frame (see Figure 8.5, right). The feedback control laws follow from Samson and Ait-Abderrahim (1991) as

$$\begin{aligned}
 \omega &= \omega_{\text{ref}} - \frac{k_3}{k_2} \epsilon_\theta + \frac{k_6}{k_2} \epsilon_x \\
 &\quad - \frac{k_1}{k_2} \left(\delta_x \omega_{\text{ref}} \frac{\cos(\epsilon_\theta) - 1}{\epsilon_\theta} - v_{\text{ref}} \frac{\sin(\epsilon_\theta)}{\epsilon_\theta} \right) (\epsilon_y + \delta_x \epsilon_\theta), \\
 v &= v_{\text{ref}} + k_3 k_5 \epsilon_x \\
 &\quad + \left[2 k_3 k_4 + \left(v_{\text{ref}} \frac{\cos(\epsilon_\theta) - 1}{\epsilon_\theta} + \delta_x \omega_{\text{ref}} \frac{\sin(\epsilon_\theta)}{\epsilon_\theta} \right) + k_6 \right] \epsilon_\theta \\
 &\quad + [(1 - k_1) \epsilon_y - k_1 \delta_x \epsilon_\theta] \omega,
 \end{aligned} \tag{8.6}$$

where v_{ref} and ω_{ref} are the linear and angular velocities of the reference bike, and δ_x is the control point offset given by the distance along the local x -axis from the rear wheel. In addition, the control gains can be tuned empirically under the following conditions: k_1 and k_2 are positive real numbers, k_3 , k_4 and k_5 are positive scalars assumed to be constant, $0 \leq k_4^2 < k_5$, and k_6 is any real scalar. The state vector $\mathbf{x}_{\text{track}}$ is guaranteed to converge to zero, so

long as v_{ref} and ω_{ref} are differentiable for all $t \geq 0$, these derivatives remain bounded, and the reference bike does not stop moving.

Since a 2D control strategy is used, any 3D pose feedback needs first to be transformed into the control frame in \mathbb{R}^2 before it can be utilized. This is done by finding the nearest way points at positions \mathbf{p}_i and \mathbf{p}_j in the 6-DoF path in \mathbb{R}^3 , and then using their corresponding surface normals to project the position and orientation vectors into the associated tangent plane $T_{\mathbf{p}_{i,j}}^{\text{avg}}\mathcal{S}$, in order to find the approximate offsets or offset estimates $[\tilde{c}_x \quad \tilde{c}_y \quad \tilde{c}_\theta]^T$.

8.3 Sparse Point Cloud Planning

The main objective of the second navigation solution again is to generate feasible and preferably optimal paths between two given poses on a curved surface from an input point cloud, but now relaxing the requirements for the point cloud’s regularity and density. The dense point cloud planning method avoids the additional cost of mesh generation, however still relies on a dense and rather uniform representation of the point cloud. We have used token reduction and densification by dense voting in the tensor voting framework above to construct regular environmental feature maps. Sparse point cloud planning aims at removing the equalizing and densification steps from the planning method, and instead plans paths by more directly using sparse point clouds.

We preprocess the input point cloud by sparse voting. However, instead of dense voting and the generation of environmental feature maps, quadric surface patches are fitted to the augmented point cloud. We developed our own quadric fitter, which can account for additional information, such as sensor pose or robot geometry (for an example, see Figure 8.7, left). Related approaches for fitting quadrics to point clouds can be found in the works by Vaskevicius et al. (2010); Vona and Kanoulas (2011). The second navigation solution can now be realized according to one of two different schemes: in a two-step sequential or an incremental manner.

Under the *sequential planning scheme*, first an initial discrete path is planned based on the point cloud only, by making strong use of nearest neighbor queries. We apply an adapted RRT* algorithm, which connects discrete poses within a well-specified ball radius, and hence plans paths inside a narrow band around the surface (for an example, see Figure 8.7, center). The original RRT* algorithm is described by Karaman and Frazzoli (2011); alternative related methods like the algorithms presented by Mémoli and Sapiro (2001, 2005) could be applied instead. After initial path generation,

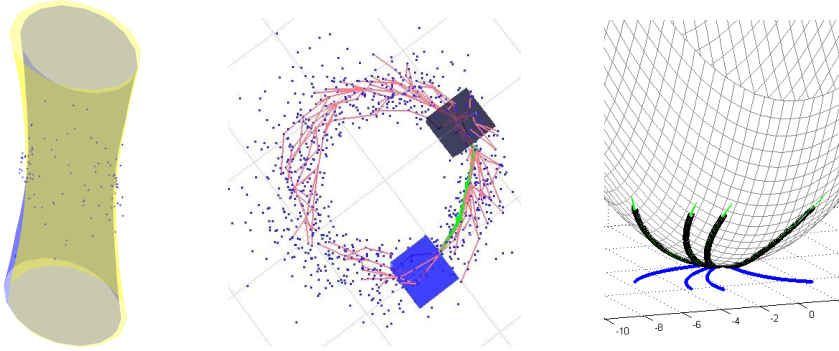


Figure 8.7: Sparse point cloud planning. Left: Fitting a quadric surface (yellow) to a noisy input point cloud (blue points) for locally modeling the underlying curved surface (blue). Center: Planning a discrete path (green) with RRT* from a start region (dark gray) to a goal region (blue) by sampling from the input point cloud (blue points). The expanded edges of the tree are shown in red. Right: Developing motion primitives on a quadric, in order to connect waypoints along the path with feasible motions. The robot positions are shown in black and the robot’s heading vector in green; blue is the projection of the trajectories to the 2D plane.

the discrete poses are connected by fitting quadrics to the point cloud and extending motion primitives on these quadrics, as we move along the surface (for an example, see Figure 8.7, right). Note that motion primitives have mostly been designed for systems with workspaces of \mathbb{R}^2 or \mathbb{R}^3 . Generating motion primitives for a robot moving on a curved surface is more evolved; not only a model of the robot but also a suitable representation of the underlying surface is required.

Under the *incremental planning scheme*, fitting of quadrics and extension of motion primitives are executed incrementally, i.e., a quadric is fitted, the robot moves on the quadric by tracking a developed motion primitive and repeats the procedure as soon as it reaches the boundary of the validity area of the quadric. Our currently developed incremental path planning algorithm relies on the ARAE* algorithm (Gonzalez and Likhachev, 2011), which uses equivalence classes to eliminate the dependence on regular grids, and thus allows for sparse edge expansion; the number of extended motion primitives is controlled via the equivalence classes and the paths are driven toward the goal pose via a search heuristic.

The point clouds are obtained from measurements with a laser range finder. Hence, points of the point cloud represent real points in the environment, which are distributed around the exact physical surface according to sensing noise and the sensor characteristics. If a sampling-based planner like the RRT* is used, points can be sampled by the planner either as new points, without direct physical meaning, from the neighborhood of the set of measured points or from the set of measured points themselves. In the later case, it is interesting to think of the laser range finder being in the role of the planner's sampling routine; the laser range finder is a physical sampling unit with strong bias to sample from the underlying curved surface. The planner then just subsamples or selects best samples from the input point cloud. By adding structural information of saliency and surface normal direction to the points, the point clouds are augmented with weights in order to further improve the generated paths.

The quadrics are fitted locally, i.e., they have a limited validity area. The error of a fit can be used as a measure of the environment representation's quality for the following path planning. At the boundary of a surface patch, a new surface patch is fitted; this does not preserve global continuity but allows to establish a connection across subsequent surface patches.

The sparse point cloud planning method is still under ongoing development; current challenges include the reliable fitting of surface patches, the robust construction of discrete paths along a curved surface in the presence of noise in the input point clouds, as well as the efficient generation and tracking of motion primitives on the fitted surface patches.

8.4 Results

The proposed navigation solution is evaluated through a series of simulations and experiments with the MagneBike robot. The results are based on both synthetic, i.e., simulated and computer-generated, point cloud data, as well as 3D point clouds recorded with the rotating Hokuyo URG-04LX and Hokuyo UTM-30LX laser range finders from the real tube-like environments of Chapter 2. We first show that tensor voting is sufficiently robust to density and noise variations when applied to real point cloud data. Furthermore, we find that the planning and control algorithms for dense point cloud planning are capable of providing feasible and low-cost paths over curved surfaces. Insights into the influence of parameters, and how they can be assigned, are provided.

8.4.1 Evaluation of Tensor Voting

LIDAR point clouds suffer from noise as well as irregular and anisotropic density distributions. These characteristics have a negative influence on the tensor voting outcome, and therefore need to be investigated to ensure that meaningful results can still be obtained using LIDAR point clouds.

Identification of the Scale Parameter. Tensor voting mainly depends on a single parameter, the scale parameter σ . The scale at which the tensor voting is performed determines what types of features are detected. If σ is too large, structures will be smoothed too much and small steps or holes might be overlooked. Alternatively, if σ is too small, then noise in the point cloud, could be detected as features. Before tensor voting can be applied, an appropriate value for σ must be determined. Intuition would say that relevant features with respect to locomotion will likely occur at similar scale as the robot’s wheel size. Therefore, in our case, a good starting guess for the scale parameter σ is the wheel diameter of the MagneBike robot, $\sigma = 6$ cm. In order to determine σ experimentally, tensor voting was performed on a step feature of roughly the same height as the wheel diameter—and therefore about the scale where features become relevant to MagneBike. The edge saliencies for various σ values were analyzed. Smaller scale voting at 2 cm causes noise to trigger high edge saliencies, and large scale voting at 10 cm causes the step to be blurred over a large distance. We can see that choosing σ in the expected range of 6 cm indeed results in good performance of structure inference, and therefore a σ value of 6 cm was chosen for all the experiments presented in the following.

Robustness against Variation in Point Density. First, robustness against the variation in point density is investigated. The evaluation is carried out on two different shapes, a cylindrical tube structure and an L-shape structure, which provide examples of basic structural forms for the inspection scenario. The point clouds are computer-generated by randomly distributing points over the structures’ surface. We use the deviation in estimated surface orientations as a measure for the robustness of tensor voting. The effects of density variation can be seen in Figure 8.8, where the error in surface orientation is plotted versus the average distance between points, expressed as the relative average distance $\bar{d}_\sigma = \bar{d}/\sigma$. The error values were determined by calculating the angle between the estimated and actual orientation vectors at 6’000 sample locations on the structures. When evaluating orientation vectors, deviations in surface normals were checked where the stick saliency

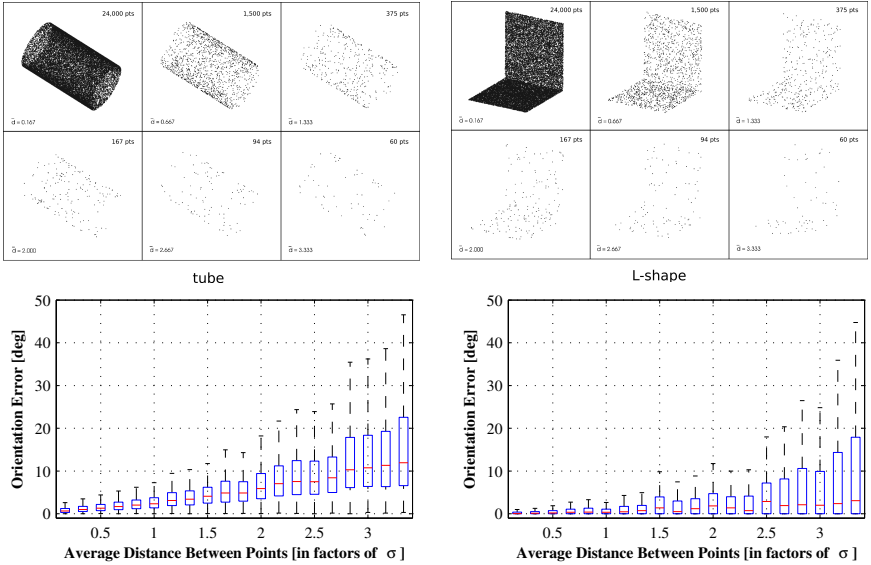


Figure 8.8: Robustness against variation in point density. The error in estimated orientations is predicted from tensor voting performed on a tube structure (left) and an L-shape structure (right) as density is varied. The results are shown in terms of relative average distance, $\bar{d}_\sigma = \bar{d}/\sigma$. A few samples of the point clouds that were used in the evaluation are shown on the top.

was dominant, i.e., not near an edge, and deviations in edge tangents where the plate saliency was dominant, i.e., at edges. It can be concluded from these plots that the error remains well behaved, even when point density drops so low that the shape is hardly recognizable. Note that, because each sample density is plotted in terms of the distance between points, the actual size of the point cloud decreases by the square of these values. We observed that typical values for the density of real LIDAR point clouds generally fall within the first two samples of highest density in the plots, with an inter-point distance d less than 2 cm, or 0.33σ respectively. Areas with much higher inter-point distances will result in low saliency values and will thus be handled implicitly as unsafe regions by the path planner.

In addition, the effects of the masking parameter β were analyzed. Point clouds with varying density were created, again using the tube and L-shape structures. These point clouds can be seen in Figure 8.9, along with graphical results of the error values, and computation time, as β is increased.

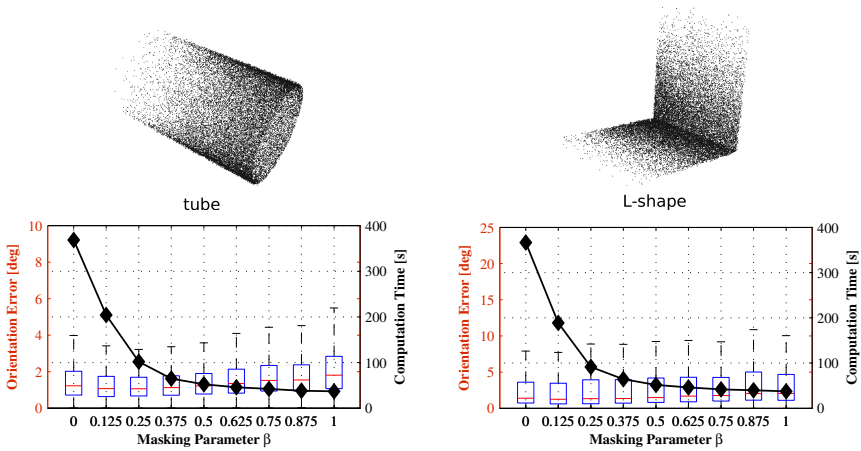


Figure 8.9: Variation of masking parameter β . The error in estimated orientations is predicted from tensor voting performed on a tube structure (left) and an L-shape structure (right). The computation time is shown as the masking parameter β is varied. Both point clouds of varying density contain 24'000 points, and are shown on the top.

These plots show that computation time of dense voting can be drastically decreased, without compromising the outcome of tensor voting. For all the tensor voting carried out in the following simulations and experiments, β is set to 0.5, due to the fact that higher values of β show little further impact on time. As stated in Section 8.2.2, this method of token reduction also unifies the point density, resulting in more uniform saliency distributions, and therefore a better representation of the actual surface.

Robustness against Variation in Noise. Next, robustness against noise variation is evaluated, using the same techniques as for the point density evaluation above. The tests are conducted on the same tube and L-shape structures, this time as noise levels are increased. The results are plotted in Figure 8.10, showing the error in surface orientation in relation to the standard deviation of the Gaussian noise distribution. The graphs illustrate the extent of noise variation used in the evaluation. In real applications, the noise depend on the sensor used and the type of reflecting surfaces. Characterization of the Hokuyo URG-04LX and the UTM-30LX laser range finders over different metal surfaces shows that the standard deviation on depth mea-

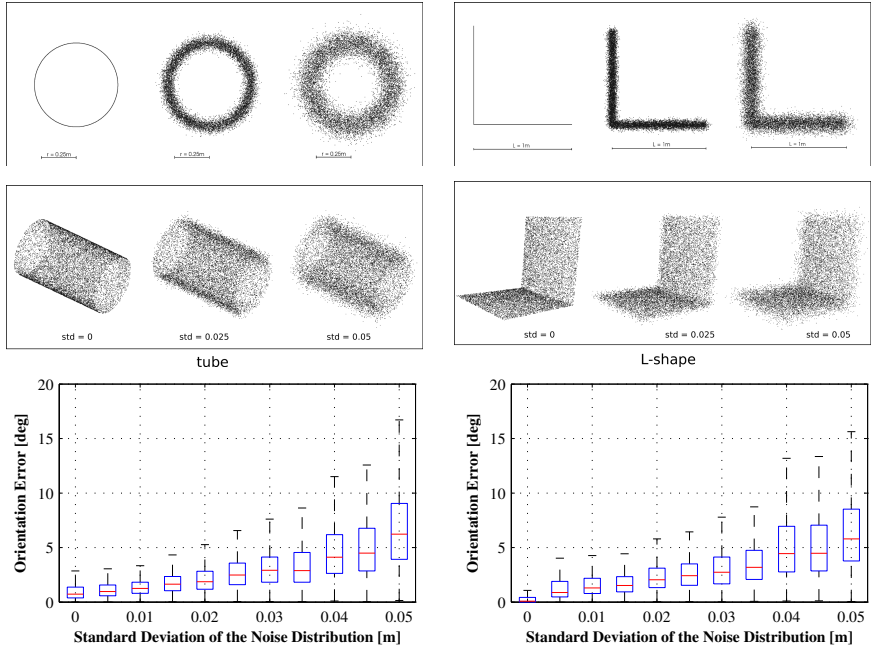


Figure 8.10: Robustness against variation in noise. The error in estimated orientations is predicted from tensor voting performed on a tube structure (left) and an L-shape structure (right) as noise is varied. The results are shown in relation to the standard deviation of the Gaussian noise distribution, which was used to generate the point clouds. A few samples of the point clouds that were used in the evaluation are shown on the top. All point clouds contain 12'000 points.

surement is 0.028 m and 0.018 m, respectively (Pomerleau et al., 2012). One can see that with the given range of noise, the orientation errors in structure inference from tensor voting remain below approximately 2.5° .

8.4.2 Evaluation of Dense Point Cloud Planning

Now that the validity of the tensor voting procedure has been established, the results can be used to evaluate the dense point cloud planning algorithms. In this section, we demonstrate that the generated 6-DoF paths are able to satisfy the mobility requirements of the MagneBike or similar inspection robots, namely the ability to climb on complex curved surfaces, overcome step obstacles in a perpendicular direction, and avoid impassable obstacles.

All the planned paths in this section are the result of processing a single point cloud, which is either a point cloud obtained from a single laser scan or assembled from several registered laser scans.

Simulation Results

Impact of Cost Functions. To start, the impact of the different cost functions is illustrated. Figure 8.11 on the left shows several paths on a tube which has a hole-type obstacle on the top part of the surface. A path must be planned from one side of the hole to the other. The cost function of Equation (8.1) with non-zero weights is used for the generation of all the four paths shown; each path emphasizes the effect of the different cost function weights. One path represents a search with high turn cost, therefore traveling through unsafe areas near the hole. Another path demonstrates the significance of the relative curvature cost; the path always prefers to travel in the plane of curvature, or in a direction where there is no curvature. A third path shows a preference for remaining on highly salient areas of the surface, avoiding the low-saliency obstacle. The weighting of these cost functions can be tuned to suit the design of the robot, nature of the environment, and the safety requirements. An example of this is shown in a path which balances the effects of each weight in order to get a safe yet efficient path.

Perpendicular Edge Traversal. An important capability of the MagneBike robots is that they are able to negotiate step-like obstacles if approached in a perpendicular direction. However, if not approached perpendicularly, proper adhesion between the magnetic wheels and the surfaces can be lost, causing MagneBike to detach from the surface. Through experiments documented by Tâche et al. (2009), the maximum angle of attack for convex steps was found to be 14° and the maximum angle of attack for concave edges is 23° . Thus, if presented with any obstacles, the MagneBike robots can either avoid or traverse them. The ability of safe edge traversal is evaluated and the results are shown in Figure 8.11 on the right. The analysis is performed on an L-shape structure, and the image shows the detected edge feature, as well as several paths traversing the edge perpendicularly. Each path was obtained using a different inflation radius for the binary edge map. The three paths correspond to increasing radii of 0 cm, 10 cm, and 20 cm. This demonstrates that, depending on the safety requirements and the size of the robot, edges can be approached from a safe distance in order to ensure successful step traversal.

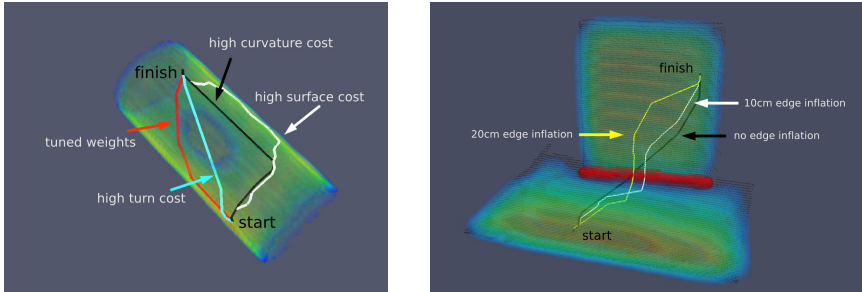


Figure 8.11: Obstacle avoidance and negotiation. Surface shading represents relative surface saliency. Left: Paths resulting from different cost weights, planned over a pipe structure with a circular hole on its upper side. The four paths were obtained for different settings of the cost function weights: One path emphasizes curvature cost, one turn cost, one surfaceness cost, and one shows a practical balance between the different weights. Right: Paths resulting from varying the edge inflation radius. The paths show 0 cm, 10 cm, and 20 cm inflations. The detected edge of the L-shape structure is highlighted in red.

Planning Paths on Geometrically Complex Surface. An example on a more complex surface with edge obstacle highlights the full abilities of the navigation solution. Figure 8.12 shows the input point cloud, the results from structure inference and feature detection, as well as the generated 3D and 2D paths. The 2D path results from the path dimensionality reduction. The robot is asked to cross to the other side of the plate safely. The navigation solution successfully generates a trajectory that steers the robot perpendicularly over the detected edge, around in the tube, mostly traveling in the plane of curvature, and again perpendicularly over the next edge.

Planning Paths through the Steam Chest. To evaluate the robustness of the path generation against real sensor noise, we use a point cloud recorded in the real steam chest environment (see Figure 2.1 in Chapter 2). This global point cloud of the steam chest highlights several challenges that a real inspection procedure may create. The exit points have a lower density than the core of the steam chest. The long middle section of the steam chest reveals the noise which is created by deformed scans and registration errors. Finally, the lower part of the middle section has a very low point density due to the scanner position and robot self-occlusion during the point cloud acquisition process. Figure 8.13 on the top presents a cut view of the same point cloud after augmentation through dense tensor voting. The lighter shades represent

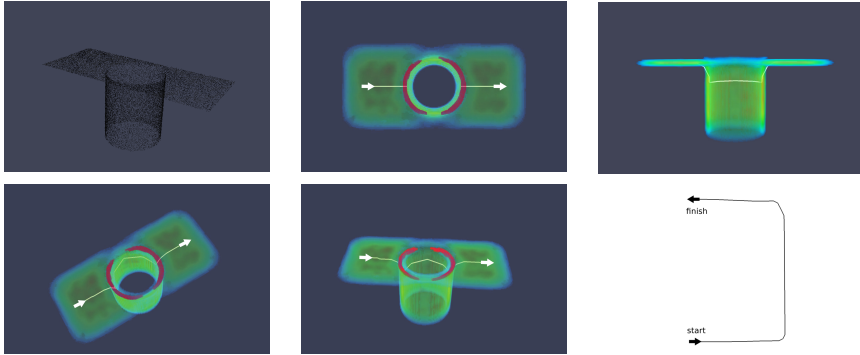


Figure 8.12: Path planning on geometrically complex surface. The input point cloud, several views of the surface with detected edge obstacle (red) and planned path (white), as well as the unwrapped 2D path are shown. The edges are crossed perpendicularly when entering and exiting the tube section.

high saliency surface regions. Note that the regions of low density are still represented but obtain a much lower saliency. The observation holds for the exit points of the steam chest as well as for the lower part of the middle section, which is less salient when compared to its upper part. Finally, the noise present in the middle section is minimized, as we can observe from the fact that the main saliency defines the shape of the pipe—even in the middle section—properly.

For the path planning, we use five of the exit points as final goals with the large opening of the steam chest as starting point. Figure 8.13 presents the resulting global paths through the steam chest. The starting point is marked with an “S”, and the five goal poses with their respective path number. The paths are properly contained within the surface, even in presence of noise and variable density. Path 1 follows the high saliency zone of the middle section on half of its length before leaving with a smooth turn to finish in the opposite orientation. Path 2 and path 3 present almost symmetric trajectories, even though the goal positions are not set to the same height. This supports that the planning is repeatable under steady environment conditions.

Figure 8.14 shows the allocation of computation time for each of the paths shown in Figure 8.13. The entire computation time is within about 30 s, which is of the same order of magnitude as the time it takes for MagneBike to record a single 3D point cloud (about 50 s). This can be considered a very usable result. The point cloud contained 32'000 points, which explains why the sparse tensor voting accounts for the majority of the computation

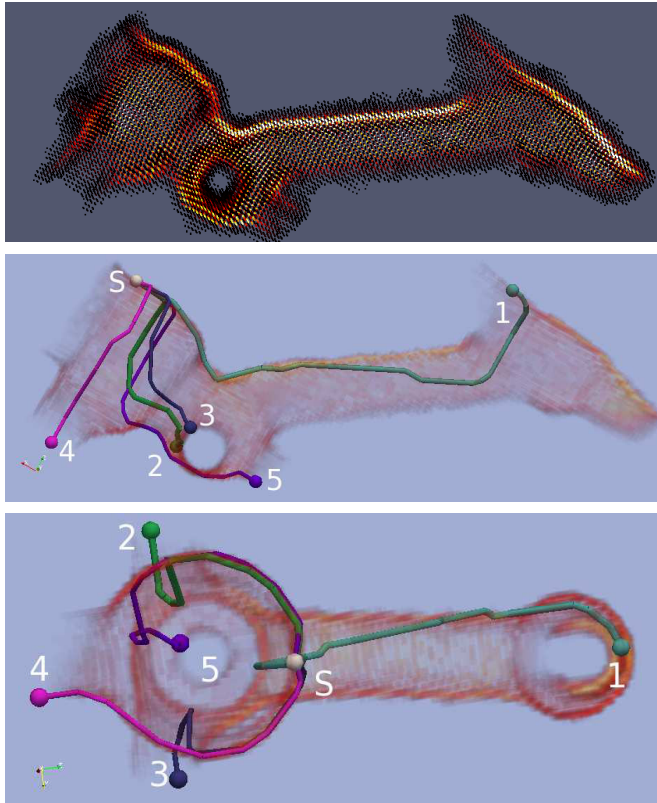


Figure 8.13: Path planning in the steam chest. Top: Saliency map after dense voting. Grid cells with very low saliency were removed for better visualization. Center and bottom: Side and top view of the five generated paths. The starting position of the robot is at the large opening of the steam chest, the goal poses are placed at five of the exit points. The global map is faded for better visualization of the planned paths.

time. After sparse voting, the number of tokens is reduced to 8'000 using token reduction (see Section 8.2.2). This drastically reduces the computation time of dense voting, despite having dense grid information at a resolution of up to 880'000 voxels with 3.5 cm side lengths. Further savings in computation time can be achieved by using an approximate range-limited nearest neighbor search in the sparse voting, and beyond that, substantial speed-ups are possible through GPU implementations of tensor voting.

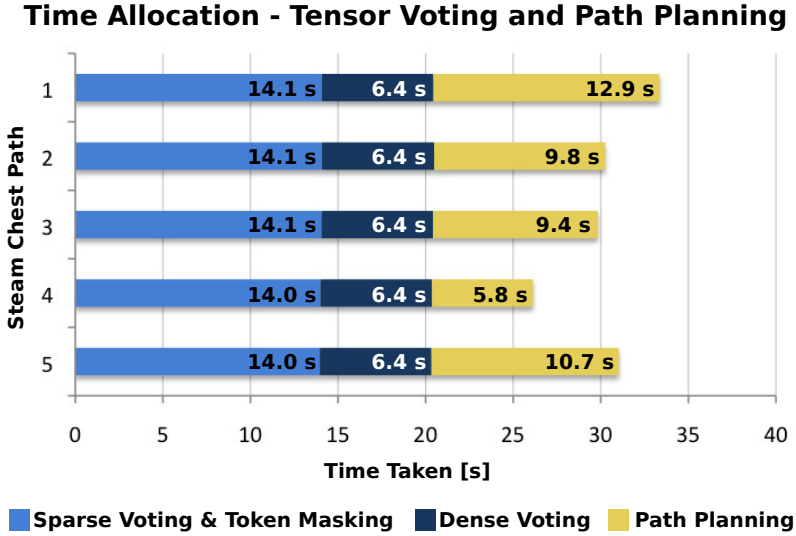


Figure 8.14: Computation times for navigation with dense point cloud planning. A breakdown of the computation times, corresponding to the individual components of the navigation solution and the paths displayed in Figure 8.13, is shown.

This experiment shows positive results in view of planning complex paths in three dimensions based on a full size, real inspection environment. Not only local paths but also paths at a more global scale can be generated successfully.

Experimental Results

In order to enhance the observations gained and further validate the feasibility of the planned paths for the execution by real robots, we performed several experiments with MagneBike in the steam chest mock-up. MagneBike plans local paths in the mock-up; obstacle avoidance and perpendicular edge traversal are demonstrated in the following.

Avoiding a Hole Obstacle. In the first experiment, the MagneBike robot is set up to navigate around a hole obstacle, similarly to Figure 8.4.2 on the left. If the hole was not there, then a simpler direct path could be chosen, which runs diagonally through the tube. As can be seen in Figure 8.15,

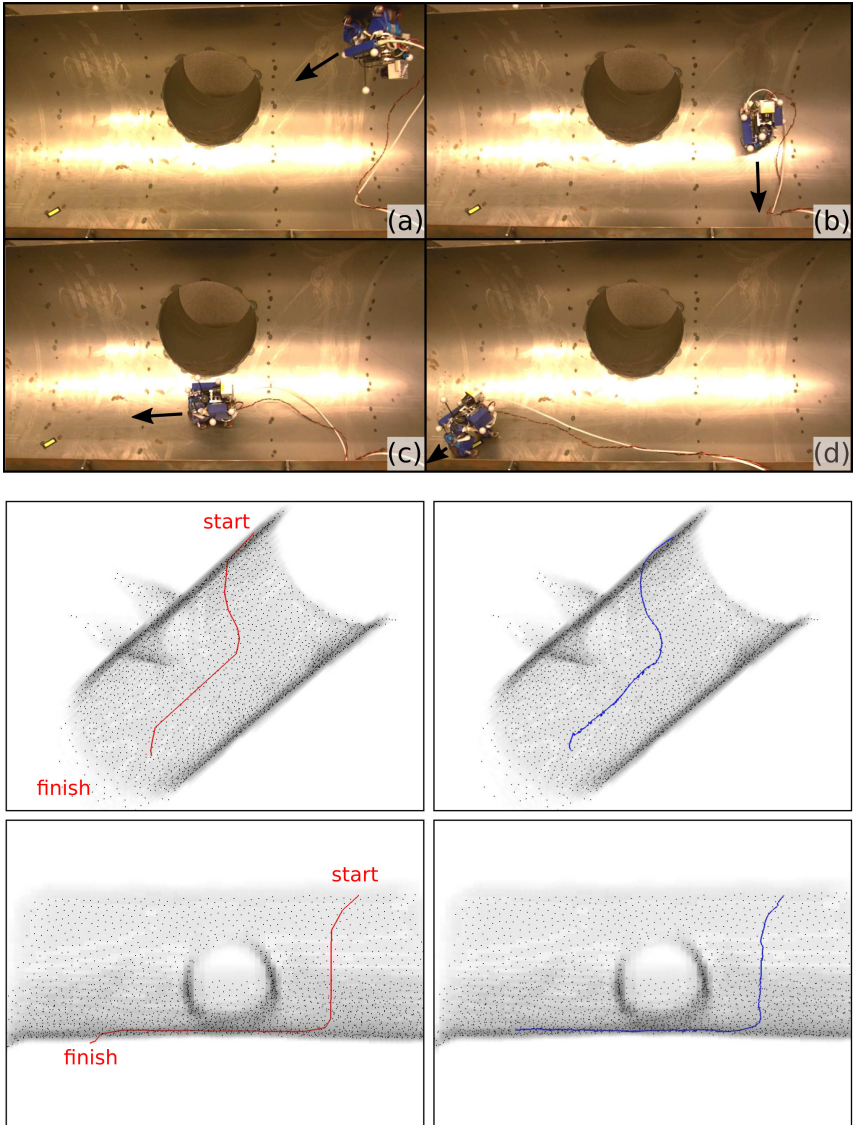


Figure 8.15: MagneBike avoiding a hole obstacle. An image sequence of the experiment is shown, followed by visualizations of the point clouds and trajectories below. The planned path is shown on the left, and the actual executed 6-DoF path is visualized on the right. The 3D point cloud contained 40'000 points.

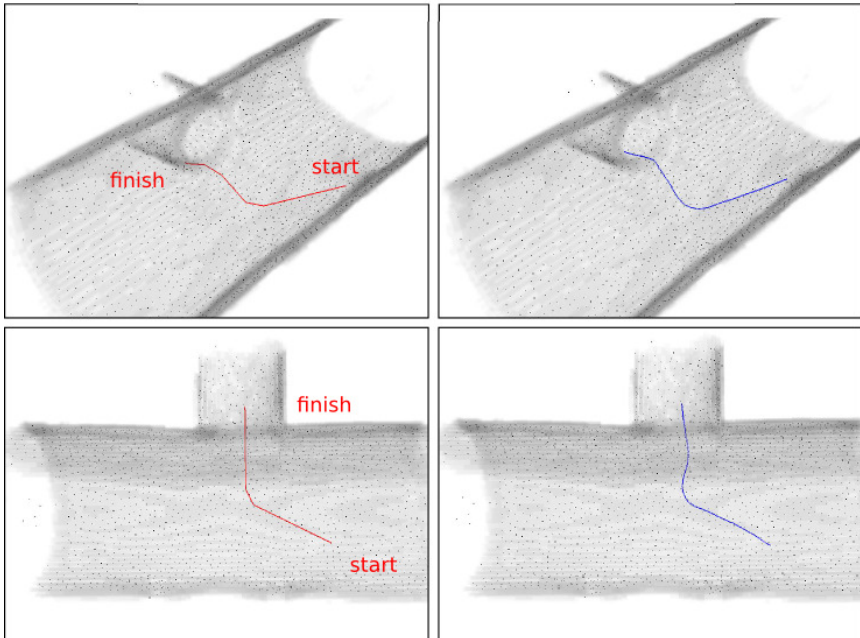
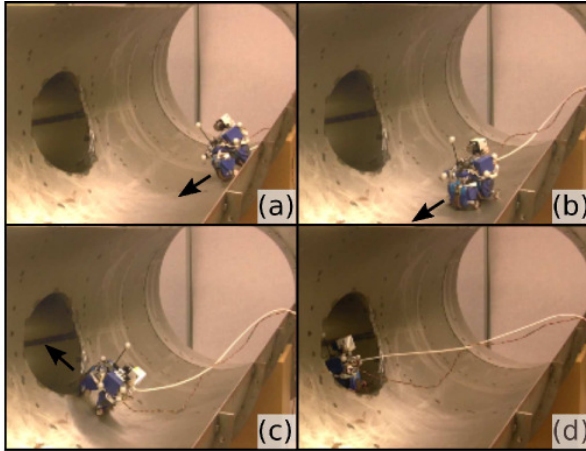


Figure 8.16: MagneBike negotiating an edge obstacle. An image sequence of the experiment is shown, followed by visualizations of the point clouds and trajectories below. The planned path is shown on the left, and the actual executed 6-DoF path is visualized on the right. The 3D point cloud contained 20'000 points.

MagneBike successfully circumvents the obstacle. The point cloud used for this experiment was recorded beforehand with a Hokuyo UTM-30LX laser range finder.

Negotiating an Edge Obstacle. The second experiment requires MagneBike to maneuver from the main pipe section of the mock-up into the small section branching off, demonstrating its ability to follow more complex trajectories with additional constraints. To be successful, the edge at the junction of the two tubes must be detected, and MagneBike must cross it perpendicularly. Figure 8.16 shows the successful result. The point cloud was again recorded with a Hokuyo UTM-30LX laser range finder.

In summary, the test results confirm the navigation solution’s capability of environment representation, 3D path planning and robot control, while avoiding or overcoming obstacles as necessary.

8.5 Summary

This chapter proposes two navigation solutions for point-based path planning on curved surfaces. In contrast to the mesh-based path planning of Chapter 7, point-based path planning allows us to forgo the additional and non-trivial processing step of mesh generation. We apply the tensor voting framework from computer vision to infer structure and geometric connectivity within the underlying point cloud. Surfaces and edges of obstacles, such as steps and holes, are successfully identified and their orientation is estimated.

Navigating a robot toward a specified goal pose requires the generation of feasible and safe paths on the surface, as well as a control method for steering the robot along the generated path. As a result of dense voting in the tensor voting process, environmental feature maps are obtained. A specialized A* graph-based planner represents the cells of the maps as a set of oriented nodes; it establishes the graph connectivity and weights of nodes in the graph incrementally, in such a way that the mobility requirements imposed by the robot and its environment are satisfied. The generated 6-DoF paths are then transformed from 3D space to 2D paths, by projecting movements into local surface planes—in a sense unwrapping the path from the curved 3D surface. A trajectory tracking controller, designed for the given kinematics of the robot, can then simply be applied in two dimensions.

As opposed to dense point cloud planning, sparse point cloud planning is concerned with planning paths more directly based on sparse point clouds. A specialized RRT* sampling-based planner incrementally samples robot states from a neighborhood within the point cloud to establish graph connectivity. Geometric surface patches, such as quadric surfaces, are fitted locally to the point cloud, and a robot moves along the surface using the fitted surface patches as local continuous representation of the environment.

The navigation solution for dense point cloud planning has successfully been tested on various 3D point clouds, including point clouds of complex curved surfaces. In addition, we demonstrated point-based path planning on the MagneBike robot, driving on the inner casing of a tube-like structure.

Chapter 9

Relative Robot Localization in 3D Space

A relative localization system is introduced in this chapter that allows for full *6D relative robot localization* in 3D space. The relative localization system consists of two complementary modules: a monocular camera module and a target module with four active or passive visual markers. The core localization algorithm running on the camera module continuously detects the marker positions in the camera image and derives the full relative robot pose in 3D space by solving the Perspective-Three-Point (P3P) problem. The system is supported by a prediction mechanism based on regression. We present results on real-world data captured by a quadrotor helicopter and from experiments with a team of two e-puck robots performing a coverage task.

Collaborative tasks in general require at least a minimum mutual exchange of data via communication or sensing. Often relative localization is sufficient for the coordination of multiple robots and absolute positioning of the robots in the world is not needed for the successful completion of a task. Collision avoidance, pattern formation, coordinated manipulation or coverage are examples where the knowledge of the relative pose provides sufficient information for planning the next actions. For instance, the hybrid coverage methods presented in Chapter 4 largely rely on the relative robot poses in order to deploy a group of robots in 2D and 3D environments.

Our relative localization system is motivated in particular by a lack of existing solutions for relative on-board localization of mobile robots moving in 3D space. Inspection robots like MagneBike, which climb walls and ceilings and perform coordinated inspection tasks, or flying robots that monitor

a certain area or hover in place for landing need compact solutions for relative 6D localization. The relative localization system is based on a simple geometric target and a monocular camera, and thus requires only a single passive sensor. This results in a lightweight system with low complexity and reduced system requirements, which enables its use on a wide range of mobile robot platforms.

Relative localization leads back to global localization if a robot measures its pose relative to a fixed landmark or another robot with known global pose. In this context, the development of the relative localization system must also be seen in close relation to vision-based test beds commonly used for ground truth and localization in multi-robot systems. Over the recent years, self-made camera-marker systems have transformed into commercial motion capture systems, which are establishing as standard localization tools in robotics.

The relative robot localization system described in this chapter has first been presented at the International Conference on Intelligent Robots and Systems (Breitenmoser et al., 2011). Section 9.1 reviews related relative robot localization systems, which are able to provide 1D up to 6D pose information. Section 9.2 introduces the theoretical background used for the P3P algorithm of our localization system. Section 9.3 presents the system's main hardware components and Section 9.4 describes the framework for continuous relative robot localization. Results of the experiments are given in Section 9.5. Section 9.6 summarizes the chapter.

9.1 Related Work

Rekleitis et al. (2002) distinguishes relative localization methods based on the level of sensed information: some relative localization systems only measure range or angle, i.e., bearing information, others the relative position, which is range and bearing, or the full pose, which besides position also includes the orientation of the observed device. Moreover, localization systems can be classified into planar or spatial systems, depending on whether 2D or 3D location information is measured.

To date still many relative localization systems assume a 2D workspace and provide 2D information of the robot's position. We include some characteristic examples for 2D space in this section but mainly focus on systems that achieve relative localization in 3D space in our review.

Navarro-Serment et al. (1999) uses beacons that emit RF and ultrasonic pulses to measure distance information between robots located in the 2D plane; three robots can jointly infer the 2D position of a fourth robot by tri-

lateration. Similarly, the Cricket indoor location system (Smith et al., 2004) consists of several beacons that sense 3D position from RF and ultrasonic pulses. Pugh and Martinoli (2006) describes a small-scale IR-based relative localization module that measures 2D positions; Roberts et al. (2012) extends the IR-based system to relative positioning in 3D space. The Wii remote, as presented by Olufs and Vincze (2009), and its sensor bar counterpart can be understood as localization modules that provide 4D information, i.e., 3D position plus roll. Combined with the built-in accelerometer, full pose information can be gained with the cost of an additional sensor.

Vision-based localization modules with single or multiple cameras are oftentimes less specific to a certain platform and need less hardware development, which results in increased flexibility for integration on existing robots. Davison and Kita (2002) uses stereo vision to detect a single marker target and measures 4D information, including the 3D position and the orientation in the 2D plane, i.e., yaw. Stein et al. (2003) obtains the 1D information of the relative distance to a target in one direction from a single camera based on the known size of the target object. Spletzer et al. (2001) uses monocular omnidirectional vision to build a localization system of similar functionality as the one reported by Navarro-Serment et al. (1999). A single robot is able to measure range and bearing information in the 2D plane. For three or more robots, the joint 3D pose of the robot team is obtained up to a scale factor.

Feng et al. (2007) presents a multi-robot system that achieves 6D relative localization. A minimum number of four heterogeneous robots is required, from which one is a climbing robot, which climbs above the three ground robots in order to track them. The ground robots themselves act as markers and form a target, applying the relative localization method of Spletzer et al. (2001). The relative pose of the climbing robot is determined by solving a P3P problem. As the target is distributed among the ground robots, visibility of the ground robots among each other and between the ground robots and the climbing robot must be guaranteed. This poses problems in confined spaces and environments with obstacles and occlusions, like the steam chest environment (see Section 2.1.1), since planning of trajectories to maintain visibility of all the robots is neither trivial nor very practical in these cases. Such environments clearly favor relative localization systems with the ability of direct robot-to-robot localization.

Eberli et al. (2011) presents a monocular vision approach that uses prior information of a target and exploits target shape and orientation to estimate up to 5 DoF of a MAV. Wenzel et al. (2011) and Masselli and Zell (2012) demonstrate another relative localization system for autonomous take-off, hovering, tracking and landing of MAVs. The system is similar in the idea to

ours, and is composed of a monocular camera and four markers. Whereas the first version solves a simplified instance of the P3P problem and additionally relies on IMU measurements for obtaining the pitch and roll angles (Wenzel et al., 2011), the latest version also finds the full 6D pose from measurements of a single camera only (Masselli and Zell, 2012).

QRTags, ARTags and the ARToolKit (Kato and Billinghurst, 1999) are alternative marker-based concepts that allow for 6D relative localization. They are primarily designed for applications in augmented reality but have been applied successfully in robotics for localization and tracking, in particular for robots moving in planar environments (also refer to the result sections of Chapter 5 and Chapter 6 in this thesis, where ARToolKit has been used). The ARToolKit provides accuracies that are comparable to the accuracies achieved with our relative localization system for tags of similar size. However, ARToolKit and similar marker-based systems are—unlike our system—not designed for omnidirectional localization in 3D space, i.e., for measuring the relative pose from (almost) any direction.

6 DoF motion capturing systems like the Vicon localization system are becoming more and more popular in the robotics community. Pintaric and Kaufmann (2007, 2008) present a low-cost pose tracking system and a target design methodology, from which we found inspiration. However, motion capturing systems are composed of multiple cameras which are installed in a fixed configuration in the environment in order to provide multiple views of the markers of the tracked targets and full coverage of the workspace. The cameras are typically equipped with IR strobe lights to illuminate the IR-reflective targets and the pose is determined from the projections to the different views. In order to obtain complete 6D pose information, a minimum number of three markers per target need to be detected in at least two cameras. In contrast, our relative localization system requires only one camera but at least four markers.

9.2 Preliminaries

The update of the camera pose in our relative localization system is computed by solving the P3P problem. This section introduces the basic theoretic concepts of the implemented solution approach to the P3P problem.

The P3P problem is the smallest instance of the Perspective- N -Point problem, which appears in camera calibration and pose estimation when determining the relative pose of a camera from N known correspondences between 2D image plane measurements and 3D world points. The P3P problem uses three such 2D-3D point correspondences to determine the camera pose

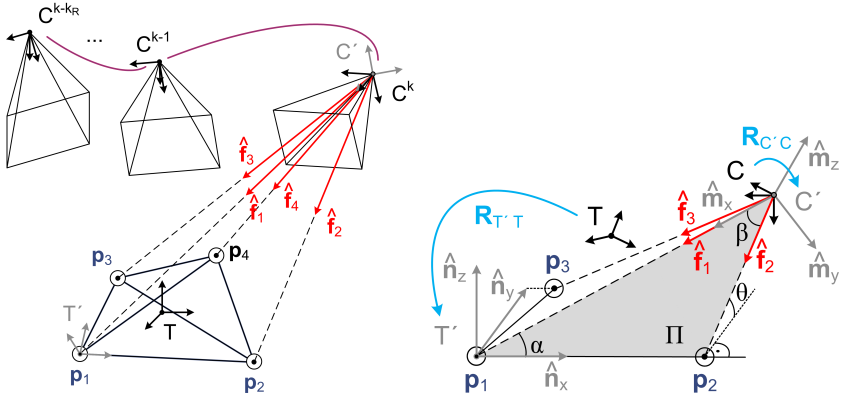


Figure 9.1: The P3P problem. Left: The pose of a moving camera, given by the sequence of camera frames $\{C^{k-k_R}, \dots, C^k\}$, is to be determined with respect to the target and the target frame T . Right: Solution approach based on the transformation T' between the intermediate camera frame C' and the intermediate target frame T' , resulting from geometric relations in the plane Π . (Adapted from Kneip et al. (2011).)

with respect to a given reference frame. This results in up to four solutions, which can be disambiguated by a fourth point. Gao et al. (2003) represents a state-of-the-art solution to the P3P problem and provides references to further relevant works on the topic.

We make use of a novel method presented by Kneip et al. (2011), and restate here the key insights. The objective is to find the pose of the camera $[\mathbf{R}_{TC} \mid \mathbf{t}_{\text{cam},T}]$. The rotation matrix \mathbf{R}_{TC} describes the camera orientation and the translation vector $\mathbf{t}_{\text{cam},T}$ describes the center position of the camera, both with respect to the target frame T . The original target frame is given by T and the current camera frame C by C^k , where $k \in \mathbb{N}$ denotes the frame at discrete time. The target is given by the four marker points at positions $\mathbf{p}_i = \mathbf{p}_{\text{tar}_i,T}$, with index $i \in \{1, \dots, 4\}$. Under the assumption of known intrinsic camera parameters, the four unitary vectors $\hat{\mathbf{f}}_i = \hat{\mathbf{f}}_{i,C}$, which point from the camera frame toward the marker points of the target, are also known. Any three of the four marker points can be used in order to solve the P3P problem; we use the first three points $\mathbf{p}_1, \mathbf{p}_2$ and \mathbf{p}_3 for the description in the following paragraphs. Figure 9.1 on the left illustrates the problem.

First we define a new intermediate camera frame C' by $\mathbf{t}_{\text{cam},T}$ and $\mathbf{R}_{C'C'}$, and a new intermediate target frame T' by \mathbf{p}_1 and $\mathbf{R}_{T'T'}$. The rotation matrices are given by the basis vectors as

$$\mathbf{R}_{C'C'} = [\widehat{\mathbf{m}}_{x,C} \widehat{\mathbf{m}}_{y,C} \widehat{\mathbf{m}}_{z,C}] \quad \text{and} \quad \mathbf{R}_{T'T'} = [\widehat{\mathbf{n}}_{x,T} \widehat{\mathbf{n}}_{y,T} \widehat{\mathbf{n}}_{z,T}], \quad (9.1)$$

where

$$\begin{aligned} \widehat{\mathbf{m}}_{x,C} &= \widehat{\mathbf{f}}_1, & \widehat{\mathbf{m}}_{z,C} &= \frac{\widehat{\mathbf{f}}_1 \times \widehat{\mathbf{f}}_2}{\|\widehat{\mathbf{f}}_1 \times \widehat{\mathbf{f}}_2\|_2}, & \widehat{\mathbf{m}}_{y,C} &= \widehat{\mathbf{m}}_{z,C} \times \widehat{\mathbf{m}}_{x,C}, \\ \widehat{\mathbf{n}}_{x,T} &= \frac{\mathbf{p}_{12}}{\|\mathbf{p}_{12}\|_2}, & \widehat{\mathbf{n}}_{z,T} &= \frac{\widehat{\mathbf{n}}_{x,T} \times \mathbf{p}_{13}}{\|\widehat{\mathbf{n}}_{x,T} \times \mathbf{p}_{13}\|_2}, & \widehat{\mathbf{n}}_{y,T} &= \widehat{\mathbf{n}}_{z,T} \times \widehat{\mathbf{n}}_{x,T}, \end{aligned}$$

and $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$. Now, the transformation between the new frames C' and T' follows from the inclination of the plane Π and further geometric relations in the plane Π , which is defined by the points \mathbf{p}_1 , \mathbf{p}_2 and $\mathbf{t}_{\text{cam},T}$ as shown in Figure 9.1 on the right. The translation vector from T' to C' is given by

$$\mathbf{t}_{\text{cam},T'} = \mathbf{t}_{\text{cam},T'}(\alpha, \theta) = \begin{bmatrix} \|\mathbf{p}_{12}\|_2 \cos(\alpha) (\sin(\alpha) \cot(\beta) + \cos(\alpha)) \\ \|\mathbf{p}_{12}\|_2 \sin(\alpha) \cos(\theta) (\sin(\alpha) \cot(\beta) + \cos(\alpha)) \\ \|\mathbf{p}_{12}\|_2 \sin(\alpha) \sin(\theta) (\sin(\alpha) \cot(\beta) + \cos(\alpha)) \end{bmatrix}, \quad (9.2)$$

and the rotation matrix from T' to C' by

$$\mathbf{R}_{C'T'} = \mathbf{R}_{C'T'}(\alpha, \theta) = \begin{bmatrix} -\cos(\alpha) & -\sin(\alpha) \cos(\theta) & -\sin(\alpha) \sin(\theta) \\ \sin(\alpha) & -\cos(\alpha) \cos(\theta) & -\cos(\alpha) \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}. \quad (9.3)$$

The angles α , β and θ are defined as indicated in Figure 9.1 on the right. β is given by $\widehat{\mathbf{f}}_1$ and $\widehat{\mathbf{f}}_2$, and is known. The reader refers to Kneip et al. (2011) for the detailed derivations. Transforming $\widehat{\mathbf{f}}_3$ into C' , the third point \mathbf{p}_3 into T' and further into C' and setting $\widehat{\mathbf{f}}_{3,C'} = \frac{\mathbf{p}_{3,C'}}{\|\mathbf{p}_{3,C'}\|_2}$ leads to two equations for the determination of the angles α and θ . Angle θ is given by the polynomial of degree four

$$a_4 \cos^4(\theta) + a_3 \cos^3(\theta) + a_2 \cos^2(\theta) + a_1 \cos(\theta) + a_0 = 0, \quad (9.4)$$

with the coefficients a_4, \dots, a_0 as described by Kneip et al. (2011). Solving for the roots of Equation (9.4) results in up to four values for $\cos(\theta)$. Angle α depends on $\cos(\theta)$ and each value for θ leads to exactly one value for α . See the work by Kneip et al. (2011) for details.

The overall transformation $[\mathbf{R}_{TC} \mid \mathbf{t}_{\text{cam},T}]$ from C to T is finally given by combining Equation (9.1), Equation (9.2) and Equation (9.3) as

$$\begin{aligned} \mathbf{R}_{TC} &= \mathbf{R}_{TT'} \mathbf{R}_{T'C'} \mathbf{R}_{C'C} \\ &= \mathbf{R}_{TT'} \mathbf{R}_{C'T'}^T \mathbf{R}_{C'C}^T, \quad \text{and} \\ \mathbf{t}_{\text{cam},T} &= \mathbf{p}_1 + \mathbf{R}_{TT'} \mathbf{t}_{\text{cam},T'}. \end{aligned} \tag{9.5}$$

We will use Equation (9.5) in Section 9.4 to find the relative pose between both the camera and the target frames.

9.3 System Overview

The 6D relative localization system is vision-based and consists of a camera and a target module. The modules are flexible and allow for customization into many directions. The following two sections present the two complementary modules and their basic mode of operation, as well as the optimization procedure of the target design.

9.3.1 Camera and Target Modules

The *camera module* detects the target module and computes the 6D relative pose of the camera module with respect to the detected target. The camera module is composed of a monocular camera and a computing device for image processing and pose estimation. Basically, any calibrated camera can be used with the camera module for target detection. However, for localization in close to planar settings, omnidirectional catadioptric cameras are the preferred choice due to their 360° FOV in the direction parallel to the image plane. Fish-eye lenses with up to 190° FOV in the direction of the camera principal axis in contrast are particularly suitable in full 3D scenarios; here the target modules need to be localized when moved in free open space, e.g., on hemispherical trajectories, around the camera module. The camera is either color or monochrome, depending on the chosen target and the application environment. Figure 9.2 shows our two realizations of the camera module of the relative localization system: the module on the left consists of an IDS uEye color camera¹ with 752 × 480 resolution and 190° FOV fish-eye lens mounted on a tripod, the module on the right uses the same camera with a 150° FOV fish-eye lens mounted on an e-puck robot. The computed relative pose is the direct relative position and orientation of

¹<http://www.ids-imaging.de>

the camera module expressed in the target frame in general. If two targets were detected by the same camera module in the case of a total of three or more modules, the indirect relative position and orientation between the two target modules could alternatively be inferred.

The *target module* includes an optimized configuration of four spherical markers, all identical in size and shape. The markers have a fixed arrangement and form a target of known geometry, arranged in a tetrahedron, which serves as visual landmark for the camera module. The markers of a target are either active or passive. Active markers can be built from IR-LEDs or LEDs of a specified color. Passive markers are colored or reflective balls. Figure 9.2 includes several implementations of the target module of the relative localization system. The module on the left is mounted on a tripod and the first module on the right on an e-puck robot; both targets have active markers of diameter $b = 1$ cm, each made from three green SMD LEDs with 120° emission angle, integrated into a spherical diffuser for improving the uniform appearance of the markers. On the right, we have another two targets with passive painted markers, one of which is a smaller target with $b = 1.5$ cm and one a larger target with $b = 3$ cm.

The camera and target modules are complementary. The 6D relative pose detected by the camera module can be inverted and communicated from the camera module to the target module. If no communication devices are available with the modules, one camera and one target module must be combined to provide for mutual relative localization. The target size and geometry, the diameter of the markers as well as the camera resolution are important system characteristics, which define the maximum distance over which relative localization can still be performed in a reliable way. During the design of the system, all three characteristics must be adjusted with respect to the requirements of the final application.

9.3.2 Optimization of Target Geometries

The target has known geometry and the markers are all identical. Hence, once the markers are detected in the camera image, the identities of the markers within a target as well as the identities of several targets among each other can only be resolved by the knowledge of the markers' spatial arrangement. The target geometry is essential for the relative localization. A good design lowers the occurrence of occlusions and similarities among different target views, and contributes to the overall robustness of the target pose prediction.

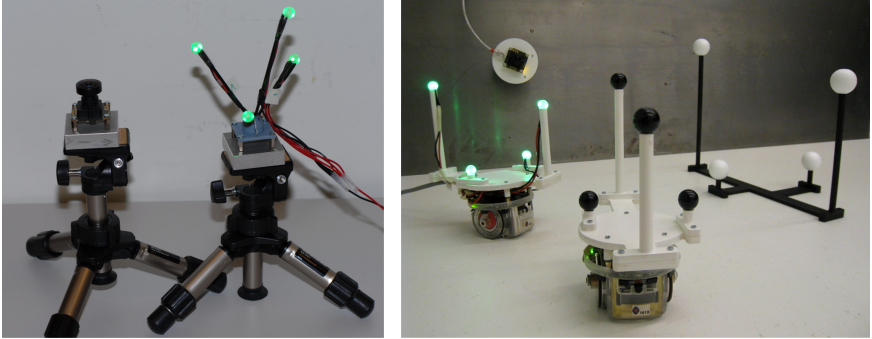


Figure 9.2: Relative localization system. Left: Camera and target modules mounted on two tripods, which can be deployed in a workspace. Right: Localization modules installed on three e-puck robots. The robot climbing the metal wall in the background is equipped with a camera module, the robot on the left features a target module with active color LED markers, and the robot in the center carries the passive version of the same target module. An additional target module with a larger passive target of the same geometry (right) represents another example of a target that can be deployed as a fixed landmark.

First, the self-similarity and symmetries in a single target must be minimized, such that they appear different if observed from different directions. This guarantees correct estimation of the target’s orientation in space. Second, if multiple targets are present in the same system, similarity among different targets must be minimized to restore the markers’ identities reliably. In addition, degenerate arrangements, such as configurations with three collinear markers or flat tetrahedrons with four nearly coplanar markers, should be avoided.

We define the similarity of target geometries according to Pintaric and Kaufmann (2008) as the smallest difference of the pairwise marker distances over all markers and targets in the system. The objective of the target optimization is to find arrangements of minimum similarity to form n targets with m markers each, i.e., the maximization of s^* , with $s^* = \min \mathcal{S}_{\text{diff}}$, where $\mathcal{S}_{\text{diff}} = \bigcup_{i \neq j} |d_i - d_j|$ with d_i and d_j the pairwise Euclidean marker distances between all the markers $i, j \in \mathcal{I}_M = \{1, \dots, nm\}$. The optimization procedure follows the procedure explained in the work of Pintaric and Kaufmann (2008) to a large part.

At the beginning, the targets are initialized by random sampling from a predefined set of discrete positions and lengths. The initial set of markers is

then optimized iteratively by maximizing the cost function

$$\mathcal{F}(\mathcal{P}, w) = \begin{cases} w s^* + (1 - w) \bar{s}, & D_1 \leq d_i \leq D_2 \wedge h_{min} \geq H \\ -G, & \text{otherwise} \end{cases} \quad (9.6)$$

where \bar{s} denotes the average distance difference over $\mathcal{S}_{\text{diff}}$, w is a free weighting parameter to balance worst-case and average target quality, D_1 and D_2 are the lower and upper bounds of the allowed pairwise marker distances and H is the minimum allowed height of the target faces. H introduces a collinearity constraint, which prevents the target to turn out too flat. G is a positive constant that penalizes violations of the constraints imposed by D_1 , D_2 and H . The dimensions of the larger target of Figure 9.2 on the right are given by the tetrahedron with vertex set $\{(0.00, -16.51, 23.66), (0.00, 13.04, 18.45), (6.26, -2.21, 5.39), (-8.02, 2.49, 6.21)\}$, and circumsphere of radius 15.2 (all given in cm). It results from an optimization with parameters set to $w = 2/3$, $D_1 = 15$ cm, $D_2 = 30$ cm and $H = 4b = 12$ cm. The smaller passive and active targets installed on the e-puck robots have the same proportions but are scaled to half the size (see Figure 9.2, right).

The actual optimization was carried out with the Nelder-Mead simplex algorithm (Lagarias et al., 1998). The optimization was repeated over 1000 runs and resulted in several best solutions due to the existence of local optima. Especially, in the case of a single target, i.e., $n = 1$, many valuable solutions remain. In order to select a final solution, the projection of the markers to the support plane tangent to the robot base was considered in a last optimization step. The target is rotated, such that the cost function in Equation (9.6) with adjusted constraints D_1 , D_2 and H is maximized. This leads to targets that are most distinct when viewed from the top, which is particularly favorable for robots climbing on opposite surfaces or flying robots that localize with respect to the ground (like in our experiments of Section 9.5 for example).

9.4 6D Relative Localization

Our relative localization system is in line with classic work in visual servoing (Wilson et al., 1996). It enables on-board relative robot localization in 3D space. We describe in this section step-by-step the processing pipeline of the relative localization algorithm that runs on top of the localization system hardware. We consider the case of a single camera with camera frame C and one visible target with target frame $T = T_l$, where $l \in \mathcal{I}_{\text{tar}} = \{1, \dots, n\}$ with $n = 1$ in our case. Figure 9.3 illustrates the sequence of operations of the proposed method.

9.4.1 Pose and Marker Prediction

The relative localization system uses single cameras to determine poses in 3D space. Poses can be predicted by tracking the markers in the image plane or the relative camera pose in 3D space. We track the pose of the camera relative to the target frame T . Pose prediction in 3D improves the robustness of the algorithm as spatial information is retained. The 3D pose estimation resolves situations with crossing marker trajectories in the image plane or markers leaving the image plane for a certain fraction of time, and allows for direct inclusion of the underlying relative motion model of the moving camera. It also allows for the application of standard pose estimation methods, such as Kalman filters.

In our implementation, we make use of a simplified approach for predicting the image coordinates of the markers of the targets. The 6D camera pose at discrete time k is estimated as $\left[\tilde{\mathbf{R}}_{TC}^k \mid \tilde{\mathbf{t}}_{\text{cam},T}^k \right]$ based on the camera pose history $\left\{ \left[\mathbf{R}_{TC}^{k-k_R} \mid \mathbf{t}_{\text{cam},T}^{k-k_R} \right], \dots, \left[\mathbf{R}_{TC}^{k-1} \mid \mathbf{t}_{\text{cam},T}^{k-1} \right] \right\}$ relative to the target frame T , where k_R is the number of regression samples taken into account. Linear regression with $k_R = 2$ already results in an accurate prediction with only slight overshooting at abrupt motion changes, as can be expected from the linear motion model. Through the modular character of our algorithm, more advanced pose estimators that take into account measurements from additional sensors—as for instance inertial readings—can easily replace the current pose prediction.

After the prediction of the 6D pose of the camera frame C relative to the target frame T is obtained, the predicted marker positions $\tilde{\mathcal{Q}}^k = \{\tilde{\mathbf{q}}_i^k\}, \forall i \in \mathcal{I}_M$, in the camera image plane follow from the projection of the target's marker positions $\mathcal{P}_{\text{tar},T} = \{\mathbf{p}_i\}, \forall i \in \mathcal{I}_M$. In the case of a pinhole camera model for example, the predicted marker positions in the camera image plane are given by $\tilde{\mathbf{q}}_i^k = K \left[\tilde{\mathbf{R}}_{TC}^k \text{ }^T \mid -\tilde{\mathbf{R}}_{TC}^k \text{ }^T \tilde{\mathbf{t}}_{\text{cam},T}^k \right] \mathbf{p}_i$, where K defines the intrinsic camera parameter matrix.

9.4.2 Blob Extraction

The predicted marker positions $\tilde{\mathcal{Q}}^k$ allow for extracting a region of interest (ROI) in the image up-front. Thus any image processing operations can be constrained to the ROI, which leads to substantial increase in speed and robustness. During the first k_R runs of the initialization phase, where the prediction is still inaccurate or unavailable, an adapted version of the blob extraction is applied to the whole image. Besides, in the case of detection

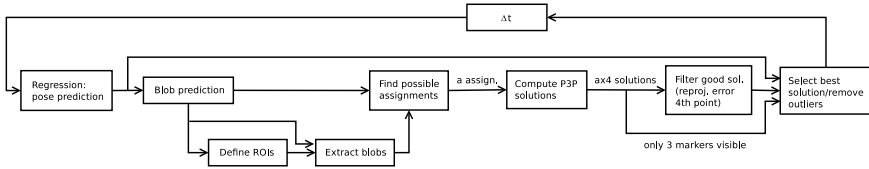


Figure 9.3: 6D relative localization. Processing pipeline of the relative localization algorithm running on the camera modules.

failure, blobs may be searched in the entire image for reinitialization.

The actual blob extraction mainly applies standard image processing methods, which makes it simple to reimplement on other platforms. The blob extraction for grayscale images works similarly for active and passive markers, only that the thresholds are computed inversely depending on whether light or dark markers are used. First, the image foreground is extracted by iterated thresholding and filtering (see Figure 9.4, left). A strategy based on successive application of two adaptive thresholds has shown to be especially useful for initial blob detection over the entire image, or for scenes with changing light. The first threshold λ_1 is computed after Otsu’s criterion and assumes that the image contains clear fore- and background. Whereas threshold λ_1 locates the blobs in the image, threshold λ_2 refines their shape and location by being less restrictive. λ_2 exploits the fact that there are much more background pixels than foreground pixels. A Gaussian is fitted to the grayscale histogram of the image and λ_2 is defined as a multiple of the standard deviation. After thresholding with λ_1 , the extracted set of blobs is filtered by executing the first round of outlier rejection: too small and too large blobs as well as blobs with strong nonconvexity and excentricity are removed. The window that encloses all the remaining blobs is subject to threshold λ_2 , and a second round of outlier rejection is started.

If the number of blobs still exceeds the specified number of nm markers in the system, the nearest neighbor of each blob is computed to subsequently remove the blobs with farthest distance to all the remaining blobs, or to merge the two closest blobs respectively. An alternative would be to make use of the prediction of the blob position in the image to reject false positives. On the other hand, if the number of detected markers is below nm , a cascade of recovery methods with increasing complexity—from simple erosion to circular Hough transform (applied to the unwrapped image for lenses of high distortion)—is executed. Finally, the blob centers $\mathcal{Q}^k = \{\mathbf{q}_i^k\}, \forall i \in \mathcal{I}_M$, result from the centroid calculation of the blob areas with subpixel precision.

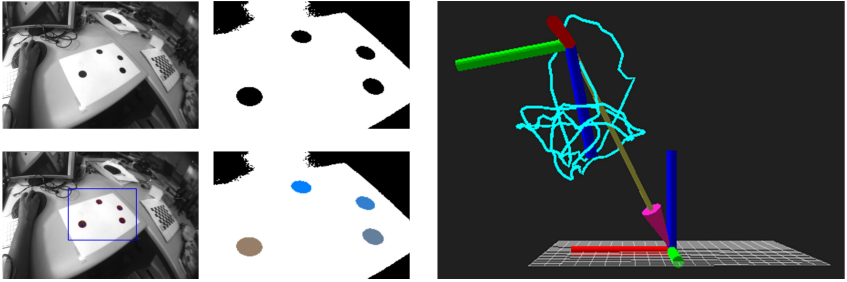


Figure 9.4: Blob extraction from planar target. Left, clockwise: Four markers of a planar target are tracked in a ROI. The image is thresholded and the blobs are detected. The four labeled blobs result from the foreground extraction and the blob centers are calculated. Right: Trajectory of the pose of a handheld camera module moved above the planar target.

We noticed that passive markers sometimes are affected by shadows on the bottom side, which has the unfavorable effect of cutting a segment of the circular blob away when thresholding. In these cases, we suggest to fit an ellipse to the (unwrapped) blob and determine the blobs' centers by intersection of the perpendicular bisectors of two chords. A circular Hough transform could be used alternatively.

The use of color cameras in combination with colored markers rather simplifies the blob extraction step. The foreground extraction as described above for grayscale images is then replaced by thresholding the image in the YUV color space, as proposed by Bruce et al. (2000). This adds robustness to the blob detection because the relevant color information on the U and V dimensions is decoupled from marker brightness. In naturally dark environments, e.g., at night, in tunnels or pipelines, active targets are favorable. In man-made environments of changing light conditions, e.g., in indoor hallways or open industrial structures, passive color targets are most useful. A clear limitation of the approach is shown in cluttered indoor or outdoor environments of changing light and color.

9.4.3 Pose Update

After having found the marker positions \mathcal{Q}^k in the image plane, one still needs to determine their correspondences to the 3D target points. In the worst case, e.g., during initialization when no prediction exists, a maximum of $(nm)! = 4! = 24$ permutations \mathcal{S}_{nm} , with $\sigma : X_{nm} \rightarrow X_{nm}$, results for a single

target with four markers ($n = 1$, $m = 4$). After initialization, this number can be strongly decreased by the pose prediction and the resulting distances of the predicted (correspondence known) and extracted (correspondence unknown) marker positions \tilde{Q}^k and Q^k in the image plane. Thus S_{nm} is restricted to a subset of permutations with high likelihood. However, there might be ambiguous situations remaining, for instance when two of the markers are close to each other.

Once all the possible permutations are determined, the final camera pose with respect to the target frame $[\mathbf{R}_{TC}^k \mid \mathbf{t}_{\text{cam},T}^k]$ is to be computed. The identical problem of finding the pose of a camera given three points in the world frame (and their corresponding points in the image plane) is well-known in computer vision as the P3P problem. We make use of the closed-form solution introduced in Section 9.2, which derives translation $\mathbf{t}_{\text{cam},T}^k$ and rotation \mathbf{R}_{TC}^k of the camera with respect to the target frame T directly, i.e., without further need for intermediate derivations of first the target point coordinates in the camera frame C , and then the aligning transformation of two point groups. Increased numerical stability and especially computational efficiency is gained, which favors any lightweight implementation. As the geometry of the target and the positions of the four markers $\mathcal{P}_{\text{tar},T}$ are known, Equation (9.5) leads to four solutions for the camera position $\mathbf{t}_{\text{cam},T}^k$ and orientation \mathbf{R}_{TC}^k with respect to T . The target pose with respect to the camera frame C is inversely given by $\mathbf{R}_{CT}^k = \mathbf{R}_{TC}^{k\text{T}}$ and $\mathbf{t}_{\text{tar},C}^k = -\mathbf{R}_{CT}^k \mathbf{t}_{\text{cam},T}^k$.

In our case, the correspondences between the 2D points in the image plane and the 3D world reference points are not predefined. Each of the permutations in S_{nm} serves as starting point to solve the above P3P problem with $4!4 = 96$ solutions in the worst case. The reprojection of the fourth marker position \mathbf{q}_4^k is then used for disambiguation; the candidate transformations for which the fourth point does not fit are removed from the solution set. In the initialization phase, multiple valid hypotheses may be maintained until there is only one remaining. If not in the initialization mode, the unique solution $[\mathbf{R}_{TC}^k \mid \mathbf{t}_{\text{cam},T}^k]$ is finally obtained by selecting the remaining candidate that is closest to the prediction $[\tilde{\mathbf{R}}_{TC}^k \mid \tilde{\mathbf{t}}_{\text{cam},T}^k]$. Alternatives for outlier rejection and filtering are the inclusion of prior knowledge, such as the knowledge that a ground robot is always moving in the 2D plane, the use of robot odometry information, or the inclusion of further predictions for the same target by other camera modules.

Opposed to our assumption, markers could also be distinct. By means of different colors, color codes or varying emission frequencies of pulsed active

markers, the identities are assigned to the markers statically, independent of any 3D information. Even though 2D-3D point correspondences can be coded with the help of distinct markers, staying with the more general problem formulation of having identical markers per target leaves the possibility to use colors or emission patterns for the differentiation among multiple targets. In summary, target modules with distinct markers improve the robustness of the relative localization by simplifying the marker identification process, increasing the stability when searching for 2D-3D point correspondences and reducing the importance of optimality in the target design.

9.5 Results

The relative localization system is evaluated by three experiments: a hand-held camera module is moved over a target module placed as fixed landmark on the ground, a flying robot is localized against the same landmark, and relative localization between a ground and a climbing robot is established for a coverage task.

9.5.1 Localization of a Handheld Module

The first experiment characterizes the relative localization system and analyzes its accuracy. A camera module with a Point Grey Firefly MV monochrome camera² with 752×480 resolution and 90° FOV lens is moved above the larger target with white passive markers (Figure 9.2, right), which is positioned as a fixed landmark on the ground. The experiment is carried out in a laboratory room with a Vicon motion capture system installed, which provides the ground truth of the camera module's trajectory.

The module is moved over the target by hand to produce rich and well-controlled trajectories. The ground truth and the measured pose obtained from our relative localization system, as well as the corresponding localization errors, are shown in Figure 9.5. The system achieves following accuracy: the position error is between 0.1 cm and 12.2 cm with a mean of 1.5 cm and standard deviation of 0.7 cm; the orientation error lies between 0.1° and 4.5° with a mean of 1.2° and standard deviation of 0.4° . The distance from the camera of the camera module to the target module varied between 67.7 cm and 174.1 cm during the experiment. The bias in the errors may originate from small inaccuracies in the initial calibration against the ground truth or from slight deviations of the actual target geometry.

²<http://www.ptgrey.com>

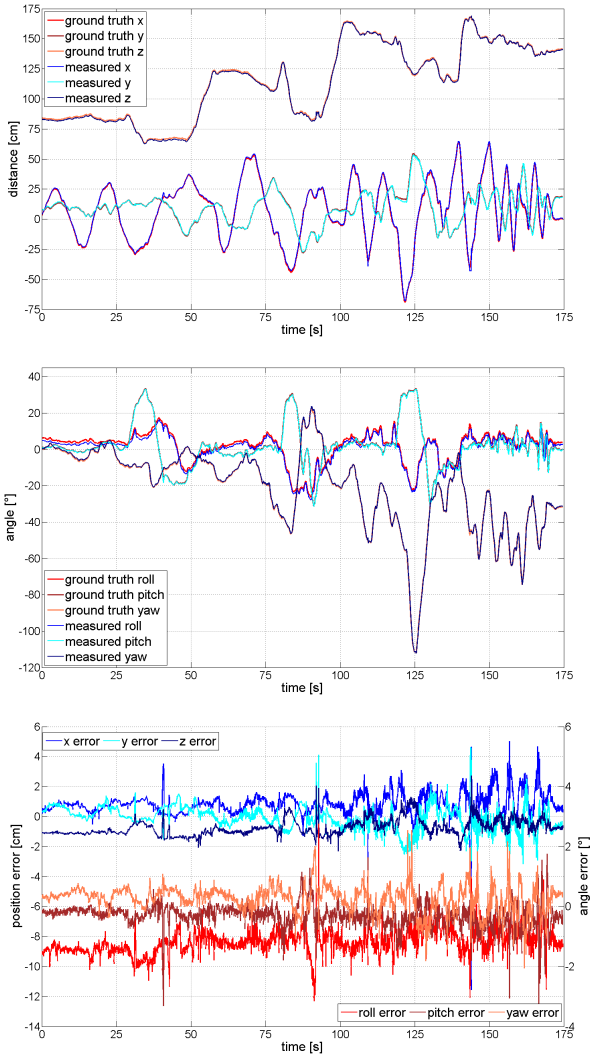


Figure 9.5: Relative localization of a handheld module. A camera module is moved manually over a fixed target module. The 6D pose, i.e., position and angles, measured from the localization system (blue), the ground truth (red), and the corresponding errors with respect to the ground truth are shown.

9.5.2 Aerial Vehicle Localization

We mounted the camera module from the first experiment on the bottom of an AscTec Hummingbird quadrotor³, pointing toward the ground. The quadrotor was flown over the target up to a height of 3 m. At this distance, the localization accuracy is 2.2 cm and 1.0° in average. The position error is between 0.1 cm and 19.5 cm with a mean of 2.6 cm and standard deviation of 1.9 cm; the orientation error lies between 0.1° and 5.2° with a mean of 1.0° and standard deviation of 0.4°. The distance from the camera on the quadrotor to the target module varied between 89.5 cm and 302.1 cm over the experimental run. The 3D trajectory of the flying quadrotor is visualized in Figure 9.6.

In both, the first and this second experiment, we observed only a minor increase in the localization error when the distance between the camera and target module was increased to the maximum range. However, the relation of the distance between modules, the target and marker sizes as well as the camera resolution define limitations of the relative localization system. For example, at a distance of 3 m, the size of the detected markers is reduced to only a few pixels for the used target and camera with 752×480 resolution.

In the experiments, several situations occurred where only two or three of the four markers were visible in the camera for a certain duration. The localization algorithm proved to be robust and recovered from these situations, even though the accuracy is affected. The maximum errors of 12.2 cm and 4.5° from the first experiment, for instance, are both caused by situations with only three markers visible in the camera image.

9.5.3 Relative Localization for Multiple Robots

The third experiment tests the relative localization modules for their application in a multi-robot system. The experimental setup comprises an active and passive target module as well as two camera modules. The first camera module consists of a Point Grey Flea2 color camera with 1280×960 resolution installed overhead at a height of 230 cm. It remains static throughout the duration of the experiment and can be thought of as representing a robot, which is aware of its absolute position at the ceiling (e.g., it might have means for climbing and performing global localization). The passive or active target module, respectively, is fitted to robot r_1 , an e-puck robot that sweeps the surface on the ground with a back-and-forth sweeping pattern of dimen-

³<http://www.ascotec.de>

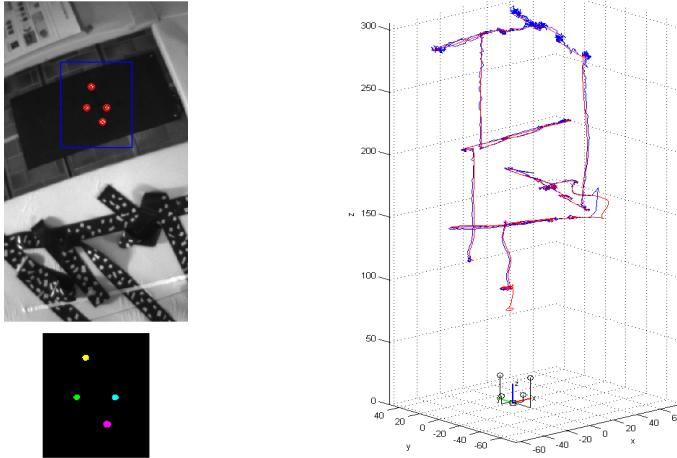


Figure 9.6: Relative localization of a flying quadrotor. Left: Tracked landmark in the quadrotor’s view. Right: The ground truth (red) and measured (blue) trajectories are visualized (positions given in cm).

sions $30 \text{ cm} \times 42 \text{ cm}$. The second camera module uses an IDS uEye color camera with 752×480 resolution and 150° FOV lens, which is mounted on a second e-puck robot, robot r_2 (see also Figure 9.2, right). Robot r_2 is further augmented with magnets in the structure, such that it can climb the wall and sweep the inclined surface by a back-and-forth sweeping pattern with dimensions $25 \text{ cm} \times 21 \text{ cm}$.

Figure 9.7 shows the two e-puck robots during the coverage task and the views of the camera modules, as well as the view from an additional external camera recording the experiment. The ground truth coverage paths and the actual coverage paths are projected into the image planes. The measured paths deviate from the ground truth by a few centimeters. The localization accuracy of the target module however varies significantly between robot r_1 and robot r_2 . The average and maximum position errors are 1.0 cm and 3.3 cm for robot r_1 and 1.7 cm and 7.8 cm for robot r_2 . Whereas the overhead camera features a higher resolution and remains completely static during the experiment, robot r_2 carries a camera with lower resolution and moves itself while detecting the moving target module on robot r_1 . Furthermore, the trajectory of robot r_2 is transformed via robot r_1 into the frames of the overhead camera and the external camera, which results in an addition of localization errors. The same setup was tested for robot r_1 equipped with

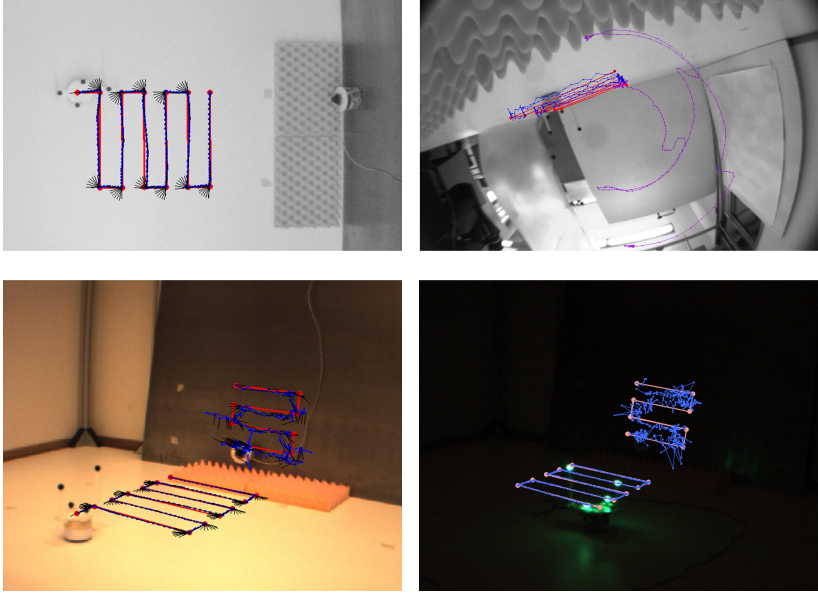


Figure 9.7: Relative localization of multiple robots (ground truth paths in red, measured paths in blue, robot orientations in black; note that the measured paths are unfiltered and represent the raw data including outliers.). Top row: View from the overhead camera and view from robot r_2 , which is climbing the wall (the trajectory in violet is obtained from projecting the robot position to the image plane over the entire experiment, applying the momentary transformation of each time step). Bottom row: View from the external camera for robot r_1 equipped with the passive and active target modules.

the active target in a dimmed room with similar results (see Figure 9.7, bottom right).

9.6 Summary

This chapter addresses the problem of localizing the 6D poses of robots that move in full 3D space, such as flying and climbing robots, which involves vertical positions and upside down orientations. The proposed relative localization system is composed of a target and a camera module, which offer a simple, lightweight and portable solution, relying on monocular vision, with

the potential to perform relative localization in real-time. The position and orientation information between target and camera modules is determined from continuously detecting the markers in the camera image and solving a P3P problem. The relative localization system was successfully tested in physical experiments and the evaluation of the measured poses resulted in centimeter/degree accuracies, which is sufficient for most robot applications.

Regarding multi-robot systems, the relative localization system can be applied and extended in many ways. The relative localization system supports the coordination between robots and provides the relative pose as it is assumed in the coverage solutions presented in Chapter 5 and Chapter 6. The coverage algorithm of Chapter 5 requires the relative position of neighboring robots, the coverage algorithms of Chapter 6 need the relative 6D pose for sharing and merging the robots' mesh maps. Moreover, relative poses are required for multi-robot collision avoidance; the relative localization system could be used in combination with reciprocal collision avoidance methods (Alonso-Mora et al., 2012a, 2013). The relative localization system can further be used to study cooperative localization and distributed pose estimation techniques (Rekleitis et al., 2002).

The relative localization system allows for multiple targets and cameras. However, especially the case with multiple target modules seen by one camera module needs further investigation. It would be interesting to see for how many different targets with identical markers the correspondence problem can still be solved to disambiguate the targets. Results by Pintaric and Kaufmann (2008) are promising. Otherwise, alternatives with more than four markers per target or color coding of different targets could help.

The relative localization module has also been developed with respect to applications of the inspection scenario. Industrial structures do typically not include rich visual features and the environments are not illuminated. Target modules with colored or active markers could offer a practical solution.

Chapter 10

Conclusion

In this thesis we have studied multi-robot coverage and path planning for the inspection of curved surfaces.

The challenges of an inspection scenario are diverse: robots navigate in full 3D environments and are constrained through various obstacles and curved surfaces of arbitrary geometries. Curved surfaces can more generally be seen as examples of curved spaces in differential geometry. The inspection task is a high-level task, which comprises many of the core tasks in mobile robotics: an inspected environment must be explored and mapped, searched for single defects or covered completely, and afterwards be revisited and monitored periodically. Coverage and path planning are key components for a single mobile robot as well as for a multi-robot system on the way to achieve this task.

The developed coverage and navigation solutions apply generally. The coverage solutions deploy a group of robots in an environment with obstacles or on a curved surface. The robots collectively partition the environment and each robot is responsible for a region to fulfill a task. For example, the robots cover their assigned region completely by a sweeping pattern in the search for defects in the surface. The navigation solutions support the coverage solutions on a lower level. They plan transitioning as well as covering paths for the individual robots, such that the robots reach their next waypoint on the curved surfaces during deployment or sweeping.

In the remainder of this chapter, we first discuss our objectives and contributions in Section 10.1 and finally conclude in Section 10.2 with an outlook on future work.

10.1 Discussion of Contributions

In the introductory chapter, we have formulated one long-term vision and two more specific short-term goals. Our overall vision is to work toward autonomous mobile robots that serve as assistive inspection tools for robotic inspection of infrastructures. On the route to this goal, we agreed that the planning and coordination of actions are, besides localization and mapping, key components to enable the reliable, efficient and flexible inspection by a mobile robot system.

Path Planning for Complex Environments. We developed two navigation solutions, which are based on two different environment representations and both enable a mobile robot, in particular a climbing robot, to move on uneven terrain or arbitrary curved surfaces of a realistic 3D environment. The navigation solutions include environment modeling, path planning and robot control. The first solution is mesh-based. A triangle mesh is generated, the triangle strip planner plans 6-DoF transitioning and covering paths on the mesh, and the vector field controller steers the robot along the triangle strip path. The second solution is point-based. A point cloud is augmented by tensor voting, the dense or sparse point cloud planner plans a 6-DoF transitioning path along the point cloud, and the robot is controlled to follow the discrete path. Additional constraints, such as perpendicular approaching of an obstacle for obstacle negotiation, are considered by the planner. The navigation solutions were implemented and tested in simulations and experiments. Several mesh generation methods were evaluated and the tensor voting framework was applied to robot path planning.

Follow-up work includes the testing of the first navigation solution for mesh-based path planning with the MagneBike robots as well as the completion of the sparse point cloud planning method. Section 10.2 points to further directions of future work.

Toward Multi-Robot Coverage and Inspection. We used the hybrid coverage concept for the design of several multi-robot coverage algorithms, such as the hybrid surface area coverage. By combining the original Voronoi coverage method with path planning, nonconvexities in the environment can be considered. We developed two coverage solutions to perform Voronoi coverage on discrete representations. Multiple robots were deployed into discrete partitions over a curved surface, which is modeled by a triangle mesh. The robot deployment can furthermore adapt to local anisotropy, which can be defined by surface curvature or from user input. Once the

robots are deployed, each robot sweeps its Voronoi region along a covering path and thereby inspects the area of the region. The procedure can then be iterated to incrementally cover a larger environment. We designed coverage concepts and algorithms, derived theoretical formulations, and implemented and tested the coverage solutions in simulations and experiments.

As localization is important for the success of both robotic inspection as well as multi-robot coordination and coverage, we additionally investigated methods to achieve relative robot localization. We developed a 6D relative localization system which particularly takes environment characteristics, as they are typical for environments in robotic inspection tasks, into account.

The coverage solutions have been tested with e-puck robots; in future work, similar tests must be conducted with two or three MagneBikes. The discrete Voronoi coverage algorithms, which are currently based on triangle meshes, are further to be extended to work on point clouds as well. This is the equivalent to point-based planning and would perfect the study on point-based representations. See also Section 10.2 for a discussion of future works.

Assistive Inspection Tools. Regarding our vision of increased autonomy for assistive inspection tools, the proposed navigation and coverage solutions address several of the initially postulated desirable requirements of an assistive inspection tool. The navigation solutions lead to automated trajectory generation and improved repeatability of inspection paths, which offers benefits for periodic inspection. Some robustness is built in the Voronoi coverage method; coverage continues even if single robots fail. The navigation and coverage solutions allow for autonomous robot behaviors. A human operator would therefore only be required to provide the high-level control inputs. The Voronoi coverage and path planning can be adjusted by the user through weighting parameters. Thus expert knowledge, experience of critical locations, or partial knowledge of the environment geometry can be included in the automated inspection. By the use of multiple robots in the coverage procedure, inspection time can be reduced. Augmented point clouds and triangle meshes are standard data structures, which improve the visualization during inspection. Methods for environment modeling can support the interaction between a human operator and the inspection tool in cases where the tool is not visible or accessible.

In conclusion, we can say that we put in place some basic elements for the multi-robot coverage and path planning on curved surfaces and in other environments of challenging geometry, which we hope will prove beneficial for future works in robotic inspection and beyond.

10.2 Outlook on Future Work

Following extensions of our work represent interesting future research directions and/or need to be solved on the path toward more autonomous robotic inspection.

Extensions of Navigation Solutions

Incremental Mesh Generation and Path Planning. Our mesh-based and point-based path planning is rather static. An existing triangle mesh or point cloud must be extended in real-time. This ideally requires a dense SLAM framework. Dynamic point cloud updates may be achieved by ICP-based methods (Besl and McKay, 1992; Pomerleau et al., 2011; Tâche et al., 2009). Dynamic mesh updates result from incremental meshing and partial remeshing (Marton et al., 2009; Newcombe et al., 2011). The discrete paths are then updated by replanning (Koenig and Likhachev, 2002; Stentz, 1995).

Combined Mesh Generation and Path Planning. The mesh-based planning currently takes a full triangle mesh as input and then extracts a triangle strip path. It would be interesting to investigate in how far mesh generation and triangle strip planning can be combined, such that a mesh is only generated where the robot plans to go. That would lead to simultaneous incremental meshing and planning. In addition, kinematic and structural constraints could again be included at the level of discrete planning.

Sparse Point Cloud Planning. The sparse point cloud planning of Chapter 8 has not yet been completed. The single components, such as fitting of quadrics to 3D point clouds, sampling-based path planning on 3D point clouds, and construction of motion primitives on the quadric fits, have been developed. Next, they need to be put together and tested in simulations and experiments.

Collision Detection and Avoidance. The current navigation solutions assume enough clearance and do not focus on collision detection against the 3D environment. Particularly for motions of climbing robots in confined spaces like narrow pipes, collision detection needs to be added to the navigation solutions. In this context, if the path planning is used in combination with multi-robot systems, additional reciprocal collision avoidance needs to be included. Reciprocal collision avoidance of robots constrained to curved

surfaces presents an interesting topic. Our methods for the 2D case may serve as a starting point (Alonso-Mora et al., 2012a, 2013).

Extensions of Coverage Solutions

Voronoi Coverage and Multi-Robot SLAM. We assumed that the localization of the robots was given by a localization module. To incorporate multi-robot localization, Voronoi coverage is to be combined with a full multi-robot SLAM framework. In combination with our coverage solution of Chapter 6, which operates on triangle meshes, this also includes aforementioned extension to incremental mesh generation and path planning for a single robot.

Task Allocation and High-Level Task Planning. In the context of hybrid coverage methods, such as hybrid Voronoi coverage, it is interesting to further study schemes of the relocation of robots, the redistribution of regions and the assignment of tasks. Tasks are assigned to be completed by a robot in a region. Regions might be connected by a roadmap, such as a Delaunay graph in the case of Voronoi coverage, and the roadmap, regions and tasks can be subject to task planning and topological planning. Alternatively, region and task allocation among heterogeneous robots is a further interesting related problem.

Adaptivity, Optimality and Robot Metrics. Concerning the adaptive Voronoi coverage of Chapter 6, the combination and usage of the robot motion metric and the metric to construct the anisotropic Voronoi tessellation could be further analyzed. Moreover, the adaption and control of the partitions by user guidance can be further explored.

Point-Based Voronoi Coverage. As mentioned above (and previously motivated at the end of Chapter 6), discrete Voronoi coverage can be extended to work on point clouds. The point-based path planners of Chapter 8 are then used in combination with Voronoi coverage, and the mesh generation step could be avoided. In this context, the relations to clustering and probabilistic Voronoi diagrams might be investigated.

Extensions of the Relative Localization System

Disambiguation of Multiple Targets. The optimization and disambiguation of multiple targets in the same relative localization system, as well as the combination of the relative localization system with cooperative localization approaches, such as the methods suggested by Rekleitis et al. (2002), require further investigations.

Combination of Relative and Absolute Localization. Another interesting direction is the combination of the 6D relative localization system with global localization. This could lead to a heterogeneous multi-robot system, where several robots localize themselves relative to a leading robot, which is equipped with more powerful sensors that achieve reliable global localization, finally resulting in global localization information for the entire group of robots. For example, with respect to our inspection scenario, a MagneBike robot with heavy payload and equipped with a rotating laser range finder might provide the global reference for other MagneBike robots that carry only a relative localization module and a NDT sensor to cover the surface area.

Extensions toward Assistive Inspection Tools

Integration of Solutions. Even though a multi-robot SLAM framework has not directly been developed yet, many of the basic components are available. Localization, environment modeling, path planning and coverage methods need to be combined in a consistent framework and integrated on an inspection robot, such as MagneBike, in a next step.

Extended Field Testing. Once the integration of the combined solution is completed, the MagneBike robots or similar inspection robots, respectively, need to be tested extensively through field tests in realistic industrial environments. Special attention must be given to robust operation, failure recovery, integration of NDT sensors, tether handling and communication among multiple robots.

Appendix A

Proofs

A.1 P-Norms and the Parallel Axis Theorem

We analyze the form of Equation (3.30) in more detail and give a proof that the parallel axis theorem, Theorem 3.1, does not simply generalize for $p \neq 2$.

In the following, $N \in \mathbb{N}_{>0}$ is an arbitrary dimension and for a polynomial $P(\{z_i\}_{i=1}^N)$ in the components z_i of $\mathbf{z} \in \mathbb{R}^N$, we simply write $P(\mathbf{z})$.

Proposition A.1. (Polynomial Form of p -Polar Moment) *For p -norms with $p \in \mathbb{N}_{>0}$ and a bounded m -dimensional submanifold $\mathcal{M} \subset \mathbb{R}^N$, with $\rho: \mathcal{M} \rightarrow \mathbb{R}_{\geq 0}$ and volume form dV , such that $\int_{\mathcal{M}} \rho(\mathbf{q}) dV(\mathbf{q}) \in \mathbb{R}_{>0}$ and a set $\mathcal{Z} \subset \mathbb{R}^N$, the functional $J: \mathcal{Z} \rightarrow \mathbb{R}_{\geq 0}$,*

$$J(\mathbf{z}) = \int_{\mathcal{M}} \|\mathbf{z} - \mathbf{q}\|_p^p \rho(\mathbf{q}) dV(\mathbf{q}), \quad (\text{A.1})$$

is a multi-variate polynomial function of degree p in the components of \mathbf{z} , if one of the following conditions holds:

(1) p is even;

(2) p is odd and $\text{sgn}(\mathbf{z} - \mathbf{q}) := [\text{sgn}(z_i - q_i)]_{i=1}^N$ is constant and non-zero, $\forall \mathbf{z} \in \mathcal{Z}$ and $\forall \mathbf{q} \in \mathcal{M}$.

Proof. We first define for any vector $\mathbf{c} \in \mathbb{R}^N$ the polynomial $P_{\mathbf{c}}(\mathbf{x}) := \sum_{i=1}^N c_i x_i^p$. Then, we look at the two cases independently. For case (1), as p is even, it follows directly that $\|\mathbf{z} - \mathbf{q}\|_p^p = \sum_{i=1}^N |z_i - q_i|^p = P_{\mathbf{c}}(\mathbf{z} - \mathbf{q})$, with $c_i = 1, \forall i \in \{1, \dots, N\}$.

For case (2), we get $\|\mathbf{z} - \mathbf{q}\|_p^p = \sum_{i=1}^N |z_i - q_i|^p = \sum_{i=1}^N [\text{sgn}(z_i - q_i) (z_i - q_i)]^p = \sum_{i=1}^N \text{sgn}(z_i - q_i) (z_i - q_i)^p$. From the assumption that $\text{sgn}(\mathbf{z} - \mathbf{q})$ is constant, it follows that $\|\mathbf{z} - \mathbf{q}\|_p^p = P_{\mathbf{c}}(\mathbf{z} - \mathbf{q})$, with $\mathbf{c} = \text{sgn}(\mathbf{z} - \mathbf{q})$. Having $\|\mathbf{z} - \mathbf{q}\|_p^p = P_{\mathbf{c}}(\mathbf{z} - \mathbf{q})$ for both cases we get

$$\begin{aligned}
 J(\mathbf{z}) &= \int_{\mathcal{M}} \|\mathbf{z} - \mathbf{q}\|_p^p \rho(\mathbf{q}) dV(\mathbf{q}) \\
 &= \int_{\mathcal{M}} \sum_{i=1}^N c_i (z_i - q_i)^p \rho(\mathbf{q}) dV(\mathbf{q}) \\
 &= \int_{\mathcal{M}} \sum_{i=1}^N c_i \sum_{k=0}^p \binom{p}{k} z_i^k (-q_i)^{p-k} \rho(\mathbf{q}) dV(\mathbf{q}) \\
 &= \sum_{i=1}^N \sum_{k=0}^p \left[c_i \binom{p}{k} \int_{\mathcal{M}} (-q_i)^{p-k} \rho(\mathbf{q}) dV(\mathbf{q}) \right] z_i^k. \quad (\text{A.2})
 \end{aligned}$$

This proves J to be a polynomial function in z_i of degree $\leq p$.

But the z_i -monomials of degree p are $\left[c_i \int_{\mathcal{M}} \rho(\mathbf{q}) dV(\mathbf{q}) \right] z_i^p$; and by assumption their coefficients $c_i \int_{\mathcal{M}} \rho(\mathbf{q}) dV(\mathbf{q})$ are nonzero. So the degree is actually exact p .

Finally, this results in polynomials of degree p in the components of \mathbf{z} for both cases. \square

From Proposition A.1, we are now able to draw the following conclusion.

Corollary A.2. (Bounds for a Generalization of the Parallel Axis Theorem to p -Norms) *The functional J from Proposition A.1, when $p > 2$ and $m > 1$, and one of the two cases of Proposition A.1 applies, has the form*

$$J(\mathbf{z}) = J(\mathbf{0}) + C \|\mathbf{z}\|_p^p \quad (\text{A.3})$$

on a set \mathcal{Z} , where $C \in \mathbb{R}$ is a constant, only if

$$\mathcal{Z} \subseteq \mathcal{Z}_C := \{ \mathbf{z} : \mathbb{R}^N \mid P_J(\mathbf{z}) - (J(\mathbf{0}) + C \|\mathbf{z}\|_p^p) = 0 \}, \quad (\text{A.4})$$

where P_J denotes the polynomial representation of J found in the proposition. And the set \mathcal{Z}_C is piecewise polynomial with degree > 0 .

Proof. We assume $p > 2$, $m > 1$, $C \in \mathbb{R}$, such that Equation (A.3) holds on \mathcal{Z} , and one of the two conditions from Proposition A.1 fulfilled, i.e.,

- (1) p is even;
- (2) p is odd and $\text{sgn}(\mathbf{z} - \mathbf{q}) = [\text{sgn}(z_i - q_i)]_{i=1}^N$ is constant and non-zero, $\forall \mathbf{z} \in \mathcal{Z}$ and $\forall \mathbf{q} \in \mathcal{M}$.

Then from Proposition A.1 it follows that J is a polynomial function in the coefficients of \mathbf{z} given by P_J . By substituting P_J for J in Equation (A.3), Equation (A.4) follows immediately.

To prove that \mathcal{Z}_C is piecewise polynomial, it is enough to show that the functional $S(\mathbf{z}) := J(\mathbf{0}) + C \|\mathbf{z}\|_p^p$ is piecewise polynomial. And this is the case as $J(\mathbf{0})$ is a constant and thus a monomial of degree ≤ 0 and $C \|\mathbf{z}\|_p^p = C P_{\text{sgn}(\mathbf{z})}(\mathbf{z})$ is piecewise (pieces of constant $\text{sgn}(\mathbf{z})$) a sum of monomials of degree p (or $-\infty$, e.g., if $C = 0$) in the components of \mathbf{z} , which follows in analogy to the proof of Proposition A.1.

It is left to show that the piecewise degree of $P_J - S$ is greater than zero. We already found S to piecewise consist exclusively of monomials of degree ≤ 0 or p . It is thus enough to show that there exists a monomial in P_J of degree k , with $0 < k < p$. We choose $k = p - 2$, and get from Equation (A.2) in the proof of Proposition A.1

$$\left[c_i \binom{p}{p-2} \int_{\mathcal{M}} q_i^2 \rho(\mathbf{q}) dV(\mathbf{q}) \right] z_i^{p-2} \quad (\text{A.5})$$

as the only monomial candidates of degree $p - 2$. Both c_i and $\binom{p}{p-2}$ have non-zero values—the first per assumption, the latter per definition. Furthermore, it follows from integral theory that

$$\int_{\mathcal{M}} q_i^2 \rho(\mathbf{q}) dV(\mathbf{q}) > 0, \quad (\text{A.6})$$

since $q_i^2 \rho(\mathbf{q}) > 0$ holds dV -almost everywhere because $m > 1$.

So we have found monomials of degree $p - 2$ in P_J but none in S . And that makes $P_J - S$ piecewise polynomial with degree > 0 and thus $\mathcal{Z}_C = \{\mathbf{z} : \mathbb{R}^N | (P_J - S)(\mathbf{z}) = 0\}$ as well. \square

It is left to explain why Corollary (A.2) makes the generalization idea given in Equation (A.3) unusable for our application. In our setting, \mathcal{M} corresponds to $A(v)$ with $v \in V_{G_i}$, and Equation (A.3) (after translating v 's mass center $\mathbf{c}(v)$ to $\mathbf{0}$) had to hold for every v (with one C_v each) and every potential robot position \mathbf{z} .

But Corollary (A.2) would then require any subset \mathcal{Z} of the potential robot positions to be contained in many different $\mathcal{Z}_{C,v} + \mathbf{c}(v)$ (the \mathcal{Z}_C from Corollary (A.2) after moving $\mathbf{0}$ back to $\mathbf{c}(v)$)—one for every vertex in V_{G_i} , for which the Proposition (A.1) applies with $\mathcal{M} = A(v)$, which are most of the vertices, whose $A(v)$ is disjoint from \mathcal{Z} . And this is very unlikely as it would, for example, mean for a robot position to be contained in the zero set of a huge set of nontrivial polynomials (one per such vertex from V_{G_i}) hardly containing a single point.

At this point, we are left with $p = 1$ or $m = 1$. In these cases, Equation (A.3) fits only for some very special pairs of \mathcal{Z} and \mathcal{M} (in fact, for most \mathcal{M} , the functional J is not polynomial when Proposition (A.1) does not apply), what makes this generalization still unusable for most practical applications. However, providing precise descriptions for those pairs would be rather intricate; given that Corollary A.2 already rules out so many application scenarios, such further study does not seem worth the effort.

Bibliography

- N. Agmon, N. Hazon, and G. A. Kaminka. Constructing Spanning Trees for Efficient Multi-Robot Coverage. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1698–1703, May 2006.
- M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point Set Surfaces. In *Proc. of The IEEE Conference on Visualization*, 2001.
- J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley. Multi-Robot System for Artistic Pattern Formation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4512–4517, May 2011.
- J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart. Reciprocal Collision Avoidance for Multiple Car-Like Robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 360–366, May 2012a.
- J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley. Image and Animation Display with Multiple Mobile Robots. *The International Journal of Robotics Research*, 31(6):753–773, 2012b.
- J. Alonso-Mora, M. Schoch, A. Breitenmoser, R. Siegwart, and P. Beardsley. Object and Animation Display with Multiple Aerial Vehicles. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012c.
- J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots. In A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, L. E. Parker, and K. Støy, editors, *Distributed Au-*

- onomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics (STAR)*, pages 203–216. Springer Berlin Heidelberg, 2013.
- N. Amenta, S. Choi, and R. K. Kolluri. The Power Crust. In *Proc. of the ACM Symposium on Solid Modeling and Applications (SMA)*, pages 249–266, 2001.
- G. Antonelli, S. Chiaverini, and A. Marino. Decentralized Deployment with Obstacle Avoidance for AUVs. In *Proc. of the IFAC World Congress*, pages 12807–12812, 2011.
- P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar. Exact Cellular Decomposition of Closed Orientable Surfaces Embedded in \mathbb{R}^3 . In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 699–704, 2001.
- P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi. Uniform Coverage of Automotive Surface Patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.
- P. N. Atkar, D. C. Conner, A. Greenfield, H. Choset, and A. A. Rizzi. Hierarchical Segmentation of Piecewise Pseudoextruded Surfaces for Uniform Coverage. *IEEE Transactions on Automation Science and Engineering*, 6(1):107–120, Jan. 2009.
- N. Ayanian and V. Kumar. Abstractions and Controllers for Groups of Robots in Environments with Obstacles. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3537–3542, May 2010.
- J. E. Beasley. Lagrangean Heuristics for Location Problems. *European Journal of Operational Research*, 65:383–399, 1993.
- C. Belta and V. Kumar. Abstraction and Control for Groups of Robots. *IEEE Transactions on Robotics and Automation*, 20(5):865–875, Oct. 2004.
- C. Belta, V. Isler, and G. J. Pappas. Discrete Abstractions for Robot Motion Planning and Control in Polygonal Environments. *IEEE Transactions on Robotics*, 21(5):864–874, 2005.
- D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation Planning on Constraint Manifolds. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 625–632, May 2009.

- D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 3rd edition, 2005.
- 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, Feb. 1992.
- S. Bhattacharya, R. Ghrist, and V. Kumar. Multi-Robot Coverage and Exploration in Non-Euclidean Metric Spaces. In *Proc. of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, June 2012.
- A. Breitenmoser and R. Siegwart. Surface Reconstruction and Path Planning for Industrial Inspection with a Climbing Robot. In *Proc. of the International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.
- A. Breitenmoser, J.-C. Metzger, R. Siegwart, and D. Rus. Distributed Coverage Control on Surfaces in 3D Space. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5569–5576, Oct. 2010a.
- A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus. Voronoi Coverage of Non-Convex Environments with a Group of Networked Robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4982–4989, May 2010b.
- A. Breitenmoser, F. Tâche, G. Caprari, R. Siegwart, and R. Moser. MagneBike: Toward Multi Climbing Robots for Power Plant Inspection. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems: Industry track (AAMAS)*, pages 1713–1720, 2010c.
- A. Breitenmoser, L. Kneip, and R. Siegwart. A Monocular Vision-Based System for 6D Relative Robot Localization. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 79–85, Sept. 2011.
- A. Breitenmoser, H. Sommer, and R. Siegwart. Adaptive Multi-Robot Coverage of Curved Surfaces. In *Proc. of the International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Nov. 2012.
- J. Bruce, T. Balch, and M. Veloso. Fast and Inexpensive Color Image Segmentation for Interactive Robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2061–2066, 2000.

- F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009.
- W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated Multi-Robot Exploration. *IEEE Transactions on Robotics*, 21(3):376–386, June 2005.
- M. Burri, J. Nikolic, C. Hürzeler, J. Rehder, and R. Siegwart. Aerial Service Robots for Visual Inspection of Thermal Power Plant Boiler Systems. In *Proc. of the International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.
- C. H. Caicedo-Núñez and M. Žefran. Performing Coverage on Nonconvex Domains. In *Proc. of the IEEE Multi-Conference on Systems and Control*, pages 1019–1024, 2008a.
- C. H. Caicedo-Núñez and M. Žefran. A Coverage Algorithm for a Class of Non-Convex Regions. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 4244–4249, Dec. 2008b.
- G. Caprari, A. Breitenmoser, W. Fischer, C. Hürzeler, F. Tâche, R. Siegwart, O. Nguyen, R. Moser, P. Schoeneich, and F. Mondada. Highly Compact Robots for Inspection of Power Plants. *Journal of Field Robotics*, 29(1): 47–68, 2012.
- S. Carpin. Distributed Coverage while Not Being Covered. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012.
- J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *Proc. of SIGGRAPH*, pages 67–76, 2001.
- J. Carsten, D. Ferguson, and A. Stentz. 3D Field D*: Improved Path Planning and Replanning in Three Dimensions. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3381–3386, 2006.
- F. Cazals and M. Pouget. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. *Computer Aided Geometric Design*, 22: 121–146, 2005.
- H. Choset. Coverage for Robotics—A Survey of Recent Results. *Annals of Mathematics and Artificial Intelligence*, 31(1–4):113–126, May 2001.

- H. Choset and P. Pignon. Coverage Path Planning: The Boustrophedon Cellular Decomposition. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, 1997.
- D. C. Conner, A. A. Rizzi, and H. Choset. Composition of Local Potential Functions for Global Robot Control and Navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3546–3551, 2003.
- C. D. Correa, R. Hero, and K.-L. Ma. A Comparison of Gradient Estimation Methods for Volume Rendering on Unstructured Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):305–319, May 2011.
- N. Correll and A. Martinoli. Multirobot Inspection of Industrial Machinery. *IEEE Robotics & Automation Magazine*, 16(1):103–112, March 2009.
- J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage Control for Mobile Sensing Networks. *IEEE Transactions on Robotics and Automation*, 20(2): 243–255, 2004.
- J. Cortés, S. Martínez, and F. Bullo. Spatially-Distributed Coverage Optimization and Control with Limited-Range Interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005.
- A. J. Davison and N. Kita. Active Visual Localization for Multiple Inspection Robots. *Advanced Robotics*, 16(3):281–295, 2002.
- M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.
- J. Derenick, N. Michael, and V. Kumar. Energy-Aware Coverage Control with Docking for Robot Teams. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3667–3672, Sept. 2011.
- T. K. Dey and S. Goswami. Provable Surface Reconstruction from Noisy Samples. In *Proc. of the Annual Symposium on Computational Geometry (SCG)*, pages 330–339, 2004.
- P. Diaz-Gutierrez, A. Bhushan, M. Gopi, and R. Pajarola. Constrained Strip Generation and Management for Efficient Interactive 3D Rendering. In *Proc. of the Computer Graphics International Conference*, pages 115–121, 2005.

- E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
- G. Dobie, W. Galbraith, M. Friedrich, S. G. Pierce, and G. Hayward. Robotic Based Reconfigurable Lamb Wave Scanner for Non-Destructive Evaluation. In *IEEE Ultrasonics Symposium*, pages 1213–1216, Oct. 2007.
- Q. Du and D. Wang. The Optimal Centroidal Voronoi Tessellations and the Gersho’s Conjecture in the Three-Dimensional Space. *Computers & Mathematics with Applications*, 49(9–10):1355–1373, May 2005a.
- Q. Du and D. Wang. Anisotropic Centroidal Voronoi Tessellations and Their Applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, March 2005b.
- Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, Dec. 1999.
- Q. Du, M. D. Gunzburger, and L. Ju. Constrained Centroidal Voronoi Tessellations for Surfaces. *SIAM Journal on Scientific Computing*, 24(5):1488–1506, May 2002.
- J. W. Durham, R. Carli, P. Frasca, and F. Bullo. Discrete Partitioning and Coverage Control for Gossiping Robots. *IEEE Transactions on Robotics*, 28(2):364–378, 2012.
- D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart. Vision Based Position Control for MAVs Using One Single Circular Landmark. *Journal of Intelligent & Robotic Systems*, 61(1–4):495–512, Jan. 2011.
- B. Englot and F. Hover. Planning Complex Inspection Tasks Using Redundant Roadmaps. In *Proc. of the International Symposium of Robotics Research (ISRR)*, 2011.
- Y. Feng, Z. Zhu, and J. Xiao. Self-Localization of a Heterogeneous Multi-Robot Team in Constrained 3D Space. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350, Nov. 2007.
- D. Ferguson and A. Stentz. Using Interpolation to Improve Path Planning: The Field D* Algorithm. *Journal of Field Robotics*, 23(1):79–101, 2006.
- W. Fischer. *Design of Compact Climbing Robots for Power Plant Inspection*. PhD thesis, ETH Zurich, 2010. Nr. 18975.

- E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, Dec. 2005.
- P. T. Furgale and T. D. Barfoot. Visual Path Following on a Manifold in Unstructured Three-Dimensional Terrain. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 534–539, 2010.
- D. W. Gage. Command Control for Many-Robot Systems. In *Proc. of the Annual AUVS Technical Symposium (AUVS)*, volume 10, pages 28–34, 1992. Reprinted in: “Unmanned Systems Magazine”.
- A. Ganguli, J. Cortés, and F. Bullo. Distributed Coverage of Nonconvex Environments. In V. Saligrama, editor, *Proc. of the NSF Workshop on Future Directions in Systems Research for Networked Sensing*, Lecture Notes in Control and Information Sciences, pages 289–305. Springer, 2007.
- X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete Solution Classification for the Perspective-Three-Point Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, Aug. 2003.
- M. Garland and P. S. Heckbert. Surface Simplification Using Quadric Error Metrics. In *Proc. of SIGGRAPH*, pages 209–216, 1997.
- D. Gingras, T. Lamarche, J.-L. Bedwani, and E. Dupuis. Rough Terrain Reconstruction for Rover Motion Planning. In *Proc. of the Canadian Conference on Computer and Robot Vision (CRV)*, pages 191–198, 2010.
- J. P. Gonzalez and M. Likhachev. Search-Based Planning with Provable Suboptimality Bounds for Continuous State Spaces. In *Proc. of the Annual Symposium on Combinatorial Search (SOCS)*, 2011.
- M. Gopi. Controllable Single-Strip Generation for Triangulated Surfaces. In *Proc. of the Pacific Conference on Computer Graphics and Applications*, pages 61–69, 2004.
- M. Gopi and S. Krishnan. A Fast and Efficient Projection-Based Approach for Surface Reconstruction. In *Proc. of the Brazilian Symposium on Computer Graphics and Image Processing*, pages 179–186, 2002.
- M. Gross and H. Pfister. *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., 2007.

- A. Gusrialdi, S. Hirche, D. Asikin, T. Hatanaka, and M. Fujita. Voronoi-Based Coverage Control with Anisotropic Sensors and Experimental Case Study. *Intelligent Service Robotics*, 2:195–204, 2009.
- P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.
- D. Haumann, K. Listmann, and V. Willert. DisCoverage: A New Paradigm for Multi-Robot Exploration. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 929–934, May 2010.
- D. Haumann, A. Breitenmoser, V. Willert, K. Listmann, and R. Siegwart. DisCoverage for Non-Convex Environments with Arbitrary Obstacles. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4486–4491, May 2011.
- S. Hauri, J. Alonso-Mora, A. Breitenmoser, R. Siegwart, and P. Beardsley. Multi-Robot Formation Control via a Real-Time Drawing Interface. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, July 2012.
- G. Hayward, M. Friedrich, and W. Galbraith. Autonomous Mobile Robots for Ultrasonic NDE. In *Proc. of the IEEE Ultrasonics Symposium*, pages 902–905, Oct. 2006.
- M. Hebert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade. Terrain Mapping for a Roving Planetary Explorer. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 997–1002, 1989.
- H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. In *Proc. of SIGGRAPH*, pages 71–78, 1992.
- A. Howard, G. S. Sukhatme, and M. J. Mataric. Multirobot Simultaneous Localization and Mapping Using Manifold Representations. *Proceedings of the IEEE*, 94(7):1360–1369, July 2006.
- T. M. Howard and A. Kelly. Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots. *The International Journal of Robotics Research*, 26:141–166, 2007.
- L. Jaillet and J. M. Porta. Asymptotically-Optimal Path Planning on Manifolds. In *Proc. of the Robotics: Science and Systems VIII (RSS)*, 2012.

- I. Kamon, E. Rimon, and E. Rivlin. A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots. Technical report, CIS—Center of Intelligent Systems, Computer Science Dept., Technion, Israel, 1995.
- I. Kamon, E. Rimon, and E. Rivlin. Tangentbug: A Range-Sensor-Based Navigation Algorithm. *The International Journal of Robotics Research*, 17(9):934–953, Sept. 1998.
- I. Kamon, E. Rimon, and E. Rivlin. Range-Sensor Based Navigation in Three Dimensions. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 163–169, 1999.
- S. Karaman and E. Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
- H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In *Proc. of the IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pages 85–94, 1999.
- M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson Surface Reconstruction. In *Proc. of the Eurographics Symposium on Geometry Processing*, pages 61–70, 2006.
- R. Kimmel and J. A. Sethian. Computing Geodesic Paths on Manifolds. In *Proc. of the National Academy of Sciences (PNAS)*, pages 8431–8435, 1998.
- B. J. King. *Range Data Analysis by Free-Space Modeling and Tensor Voting*. PhD thesis, Rensselaer Polytechnic Institute, Dec. 2008.
- L. Kneip, D. Scaramuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2976, June 2011.
- L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature Sensitive Surface Extraction from Volume Data. In *Proc. of SIGGRAPH*, pages 57–66, 2001.

- S. Koenig and M. Likhachev. Improved Fast Replanning for Robot Navigation in Unknown Terrain. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 968–975, 2002.
- S. Koenig and Y. Liu. Terrain Coverage with Ant Robots: A Simulation Study. In *Proc. of the International Conference on Autonomous Agents (AGENTS)*, pages 600–607, 2001.
- J. J. Kuffner and S. M. LaValle. Space-Filling Trees: A New Perspective on Incremental Search for Motion Planning. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2199–2206, Sept. 2011.
- W. Kühnel. *Differential Geometry: Curves–Surfaces–Manifolds*. American Mathematical Society, 2006.
- J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.
- J.-P. Laumond, editor. *Robot Motion Planning and Control*. Springer-Verlag New York, Inc., 1998.
- S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
- S. M. LaValle and J. J. Kuffner Jr. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
- S. R. Lindemann and S. M. LaValle. Smoothly Blending Vector Fields for Global Robot Navigation. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 3553–3559, 2005.
- S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Sept. 1982.
- D. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer, 3rd edition, 2008.
- Z. C. Marton, R. B. Rusu, and M. Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3218–3223, 2009.
- A. Masselli and A. Zell. A Novel Marker Based Tracking Method for Position and Attitude Control of MAVs. In *Proc. of the International Micro Air Vehicle Conference and Flight Competition (IMAV)*, pages 1–6, 2012.

- G. Medioni, M.-S. Lee, and C.-K. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier Science B.V., 2000.
- F. Mémoli and G. Sapiro. Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-Surfaces. *Journal of Computational Physics*, 173(2):730–764, 2001.
- F. Mémoli and G. Sapiro. Distance Functions and Geodesics on Submanifolds of \mathbb{R}^d and Point Clouds. *SIAM Journal on Applied Mathematics*, 65(4):1227–1260, April 2005.
- M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Proc. of VisMath*, pages 35–57, 2002.
- N. Michael, M. Schwager, and V. Kumar. Editorial. *The International Journal of Robotics Research*, 31(12):1347–1348, 2012.
- F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a Robot Designed for Education in Engineering. In *Proc. of the Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.
- H. Moravec. Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids. Technical Report CMU-RI-TR-96-34, Robotics Institute, Pittsburgh, USA, Sept. 1996.
- P. Mordohai and G. Medioni. Tensor Voting: A Perceptual Organization Approach to Computer Vision and Machine Learning. In A. C. Bovik, editor, *Modern Image Quality Assessment*, Synthesis Lectures on Image, Video, and Multimedia Processing. Morgan & Claypool Publishers, 2006.
- P. Mordohai and G. Medioni. Dimensionality Estimation, Manifold Learning and Function Approximation Using Tensor Voting. *The Journal of Machine Learning Research*, 11:411–450, March 2010.
- R. Moser, C. Udell, and A. Montgomery. Automated Steam Turbine Straddle Root Disc Head Inspection. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, pages 513–520, 2007.
- A. Nash, S. Koenig, and C. A. Tovey. Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2010.

- L. E. Navarro-Serment, C. J. J. Paredis, and P. K. Khosla. A Beacon System for the Localization of Distributed Robotic Teams. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, pages 232–237, 1999.
- R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Oct. 2011.
- A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Probability and Statistics. Wiley & Sons, 2nd edition, 2000.
- R. Olfati-Saber. Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, March 2006.
- S. Olufs and M. Vincze. A Simple Inexpensive Interface for Robots Using the Nintendo Wii Controller. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 473–479, Oct. 2009.
- M. Pauly, M. Gross, and L. P. Kobbelt. Efficient Simplification of Point-Sampled Surfaces. In *Proc. of the Conference on Visualization (VIS)*, pages 163–170, 2002.
- M. Pavone, E. Frazzoli, and F. Bullo. Distributed Policies for Equitable Partitioning: Theory and Applications. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 4191–4197, Dec. 2008.
- M. Pavone, K. Savla, and E. Frazzoli. Sharing the Load. *IEEE Robotics Automation Magazine*, 16(2):52–61, June 2009.
- G. A. S. Pereira, L. C. A. Pimenta, A. R. Fonseca, L. d. Q. Corrêa, R. C. Mesquita, L. Chaimowicz, D. S. C. de Almeida, and M. F. M. Campos. Robot Navigation in Multi-Terrain Outdoor Environments. *The International Journal of Robotics Research*, 28(6):685–700, 2009.
- G. Peyré and L. Cohen. Surface Segmentation Using Geodesic Centroidal Tessellation. In *Proc. of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 995–1002, Sept. 2004.

- L. C. A. Pimenta, G. A. S. Pereira, and R. C. Mesquita. Fully Continuous Vector Fields for Mobile Robot Navigation on Sequences of Discrete Triangular Regions. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1992–1997, 2007.
- L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira. Sensing and Coverage for a Network of Heterogeneous Robots. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 3947–3952, 2008.
- T. Pintaric and H. Kaufmann. Affordable Infrared-Optical Pose-Tracking for Virtual and Augmented Reality. In *Proc. of the IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, pages 44–51, 2007.
- T. Pintaric and H. Kaufmann. A Rigid-Body Target Design Methodology for Optical Pose-Tracking Systems. In *Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 73–76, Oct. 2008.
- M. Pivtoraiko and A. Kelly. Efficient Constrained Path Planning via Search in State Lattices. In *Proc. of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2005.
- M. Pivtoraiko and A. Kelly. Differentially Constrained Motion Replanning Using State Lattices with Graduated Fidelity. In *Proc. of The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2611–2616, Sept. 2008.
- F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart. Tracking a Depth Camera: Parameter Exploration for Fast ICP. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3824–3829, Sept. 2011.
- F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart. Noise Characterization of Depth Sensors for Surface Inspections. In *Proc. of the International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.
- J. Pugh and A. Martinoli. Relative Localization and Communication Module for Small-Scale Multi-Robot Systems. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 188–193, May 2006.
- J. Pugh and A. Martinoli. The Cost of Reality: Effects of Real-World Factors on Multi-Robot Search. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 397–404, April 2007.

- I. M. Rekleitis, G. Dudek, and E. E. Miliotis. Multi-Robot Cooperative Localization: A Study of Trade-Offs Between Efficiency and Accuracy. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2690–2695, 2002.
- I. M. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset. Limited Communication, Multi-Robot Team Based Coverage. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3462–3468, May 2004.
- A. Renzaglia, L. Doitsidis, A. Martinelli, and E. B. Kosmatopoulos. Multi-Robot Three-Dimensional Coverage of Unknown Areas. *The International Journal of Robotics Research*, 31(6):738–752, 2012.
- C. W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. In *Computer Graphics*, pages 25–34, 1987.
- J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano. 3-D Relative Positioning Sensor for Indoor Flying Robots. *Autonomous Robots*, 33(1–2):5–20, Aug. 2012.
- R. B. Rusu, A. Sundaresan, B. Morisset, K. Hauser, M. Agrawal, J.-C. Latombe, and M. Beetz. Leaving Flatland: Efficient Real-Time Three-Dimensional Perception and Motion Planning. *Journal of Field Robotics*, 26(10):841–862, 2009.
- C. Samson and K. Ait-Abderrahim. Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1136–1141, April 1991.
- R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, 2007.
- M. Schwager, J. McLurkin, and D. Rus. Distributed Coverage Control with Sensory Feedback for Networked Robots. In *Proc. of the Robotics: Science and Systems II (RSS)*, Aug. 2006.
- M. Schwager, D. Rus, and J.-J. E. Slotine. Decentralized, Adaptive Coverage Control for Networked Robots. *The International Journal of Robotics Research*, 28(3):357–375, 2009.
- M. Schwager, B. J. Julian, M. Angermann, and D. Rus. Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks. *Proceedings of the IEEE*, 99(9):1541–1561, Sept. 2011. ISSN 0018-9219.

- H. Semat and R. Katz. *Physics*. Rinehart & Company, 1958.
- J. S. Shamma, editor. *Cooperative Control of Distributed Multi-Agent Systems*. John Wiley & Sons, 2008.
- A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking Moving Devices with the Cricket Location System. In *Proc. of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 190–202, 2004.
- M. Smith, I. Posner, and P. Newman. Adaptive Compression for 3D Laser Data. *The International Journal of Robotics Research*, 30(7):914–935, 2011.
- A. Solanas and M. A. Garcia. Coordinated Multi-Robot Exploration through Unsupervised Clustering of Unknown Space. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 852–858, 2004.
- D. Spears, W. Kerr, and W. Spears. Physics-Based Robot Swarms for Coverage Problems. *International Journal on Intelligent Control and Systems*, 11(3), 2006.
- B. Speckmann and J. Snoeyink. Easy Triangle Strips for TIN Terrain Model. *International Journal of Geographical Information Science*, 15(4):379–386, 2001.
- J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor, V. Kumar, and J. P. Ostrowski. Cooperative Localization and Control for Multi-Robot Manipulation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 631–636, 2001.
- G. P. Stein, O. Mano, and A. Shashua. Vision-Based ACC with a Single Camera: Bounds on Range and Range Rate Accuracy. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 120–125, June 2003.
- A. Stentz. The Focussed D* Algorithm for Real-Time Replanning. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1659, 1995.
- E. Stumm, A. Breitenmoser, F. Pomerleau, C. Pradalier, and R. Siegwart. Tensor Voting-Based Navigation for Robotic Inspection of 3D Surfaces Using Lidar Point Clouds. *The International Journal of Robotics Research*, 31(12):1465–1488, 2012.

- V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast Exact and Approximate Geodesics on Meshes. In *Proc. of SIGGRAPH*, pages 553–560, 2005.
- F. Tâche. *Robot Locomotion and Localization on 3D Complex-Shaped Structures*. PhD thesis, ETH Zurich, 2010. Nr. 18888.
- F. Tâche, W. Fischer, G. Caprari, R. Siegwart, R. Moser, and F. Mondada. Magnebike: A Magnetic Wheeled Robot with High Mobility for Inspecting Complex-Shaped Structures. *Journal of Field Robotics*, 26(5):453–476, 2009.
- F. Tâche, F. Pomerleau, G. Caprari, R. Siegwart, M. Bosse, and R. Moser. Three-Dimensional Localization for the MagneBike Inspection Robot. *Journal of Field Robotics*, 28(2):180–203, 2011.
- M. Tavakoli, C. Gonçalo, R. Faria, L. Marques, and A. de Almeida. Cooperative Multi-Agent Mapping of Three-Dimensional Structures for Pipeline Inspection Applications. *The International Journal of Robotics Research*, 31(12):1489–1503, 2012.
- J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.
- S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard. A Real-Time Expectation-Maximization Algorithm for Acquiring Multiplanar Maps of Indoor Environments with Mobile Robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–443, 2004a.
- S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, D. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. Whittaker. Autonomous Exploration and Mapping of Abandoned Mines. *IEEE Robotics & Automation Magazine*, 11(4):79–91, 2004b.
- R. Triebel, P. Pfaff, and W. Burgard. Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2276–2282, 2006.
- S. Valette, J.-M. Chassery, and R. Prost. Generic Remeshing of 3D Triangular Meshes with Metric-Dependent Discrete Voronoi Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):369–381, 2008.

- N. Vaskevicius, K. Pathak, R. Pascanu, and A. Birk. Extraction of Quadrics from Noisy Point-Clouds Using a Sensor Noise Model. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3466–3471, May 2010.
- M. Vona and D. Kanoulas. Curved Surface Contact Patches with Quantified Uncertainty. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1439–1446, Sept. 2011.
- I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed Covering by Ant-Robots Using Evaporating Traces. *IEEE Transactions on Robotics and Automation*, 15:918–933, 1999.
- K. E. Wenzel, A. Masselli, and A. Zell. Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle. *Journal of Intelligent & Robotic Systems*, 61:221–238, Jan. 2011.
- D. Wettergreen, S. J. Moreland, K. Skonieczny, D. Jonak, D. Kohanbash, and J. Teza. Design and Field Experimentation of a Prototype Lunar Prospector. *The International Journal of Robotics Research*, 29(1):1550–1564, 2010.
- W. J. Wilson, C. C. Williams Hulls, and G. S. Bell. Relative End-Effector Control Using Cartesian Position Based Visual Servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, Oct. 1996.
- K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, May 2010.
- B. Yamauchi. Frontier-Based Exploration Using Multiple Robots. In *Proc. of the International Conference on Autonomous Agents (AGENTS)*, pages 47–53, 1998.
- A. Yershova and S. M. LaValle. Motion Planning for Highly Constrained Spaces. In K. Kozłowski, editor, *Robot Motion and Control 2009*, volume 396 of *Lecture Notes in Control and Information Sciences*, pages 297–306. Springer Berlin/Heidelberg, 2009.
- S. Yun and D. Rus. Distributed Coverage with Mobile Robots on a Graph: Locational Optimization. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 634–641, May 2012.

Curriculum Vitae

Andreas Breitenmoser was born in Walenstadt, Switzerland, on May 12, 1983. He received his Master of Science in Electrical Engineering and Information Technology from ETH Zurich in April 2008. During his studies his interest was first in the field of microsystems technology, where he was a visiting student at the Scottish Microelectronics Centre in Edinburgh in 2006, under the supervision of Prof. Rebecca Cheung. His focus shifted on the Master's level more toward embedded systems and robotics. He conducted his Master's thesis on wireless sensor networks and smart objects at the Electronics Laboratory at ETH Zurich, under the supervision of Prof. Gerhard Tröster. He is currently a PhD student at the Autonomous Systems Laboratory at ETH Zurich, under the supervision of Prof. Roland Siegwart.

Andreas Breitenmoser's present research interests include multi-robot systems, cooperative control, computer graphics and robot motion planning for perception and actuation tasks. He is particularly interested in distributed coverage and robotic inspection, mobile and static sensor networks, and reciprocal decision making and interaction, with application in real 3D worlds.

His current research project is in collaboration with ALSTOM as industrial partner and the Distributed Robotics Laboratory (DRL) of Prof. Daniela Rus, at CSAIL, MIT. From 2009 to 2011, Andreas Breitenmoser spent three research visits at DRL, where he worked on coverage and exploration algorithms for wheeled and flying robots.

Andreas has been involved in the Swiss Confederation's Commission for Technology and Innovation (CTI) project "Highly Compact Robots for Power Plants Inspections" that aimed at transferring innovations from robotics to the power plant industry. Another research project was together with Dr. Paul Beardsley and Disney Research Zurich with focus on robotic entertainment and art.

Andreas is co-founder of Dacuda AG, a Swiss software company offering digitization technology based on real-time image processing and computer vision.

List of Publications

Journals (refereed)

- E. Stumm, A. Breitenmoser, F. Pomerleau, C. Pradalier and R. Siegwart, “Tensor Voting Based Navigation for Robotic Inspection of 3D Surfaces Using Lidar Point Clouds”, *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 12, pp. 1465–1488, 2012.
- J. Alonso-Mora, A. Breitenmoser, M. Ruffi, R. Siegwart and P. Beardley, “Image and Animation Display with Multiple Robots”, *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 6, pp. 753–773, 2012.
- G. Caprari, A. Breitenmoser, W. Fischer, C. Hürzeler, F. Tâche, R. Siegwart, O. Nguyen, R. Moser, P. Schoeneich and F. Mondada, “Highly Compact Robots for Inspection of Power Plants”, *Journal of Field Robotics (JFR)*, vol. 29, no. 1, pp. 47–68, 2012.

Conferences (refereed)

- J. Alonso-Mora, A. Breitenmoser, M. Ruffi, P. Beardsley and R. Siegwart, “Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots”, In *Distributed Autonomous Robotic Systems, The 10th International Symposium, Springer Tracts in Advanced Robotics (STAR)*, vol. 83, pp. 203–216, Springer Berlin Heidelberg, 2013.
- A. Breitenmoser, H. Sommer and R. Siegwart, “Adaptive Multi-Robot Coverage of Curved Surfaces”, in *Proc. of the 11th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Baltimore, USA, Nov. 2012.
- J. Alonso-Mora, M. Schoch, A. Breitenmoser, R. Siegwart and P. Beardley, “Object and Animation Display with Multiple Aerial Vehicles”, In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, Oct. 2012.
- F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas and R. Siegwart, “Noise Characterization of Depth Sensors for Surface Inspections”, In *Proc. of the 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, Zurich, Switzerland, Sept. 2012.

- A. Breitenmoser and R. Siegwart, “Surface Reconstruction and Path Planning for Industrial Inspection with a Climbing Robot”, In *Proc. of the 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, Zurich, Switzerland, Sept. 2012.
- S. Hauri, J. Alonso-Mora, A. Breitenmoser, R. Siegwart and P. Beardley, “Multi-Robot Formation Control via a Real-Time Drawing Interface”, In *Proc. of the 8th International Conference on Field and Service Robotics (FSR)*, Matsushima, Japan, July 2012.
- J. Alonso-Mora, A. Breitenmoser, P. Beardsley and R. Siegwart, “Reciprocal Collision Avoidance for Multiple Car-like Robots”, In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 360–366, St. Paul, USA, May 2012.
- A. Breitenmoser, L. Kneip and R. Siegwart, “A Monocular Vision-based System for 6D Relative Robot Localization”, In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 79–85, San Francisco, USA, Sept. 2011.
- J. Alonso-Mora, A. Breitenmoser, M. Ruffli, R. Siegwart and P. Beardley, “Multi-Robot System for Artistic Pattern Formation”, In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4512–4517, Shanghai, China, May 2011.
- D. Haumann, A. Breitenmoser, V. Willert, K. Listmann and R. Siegwart, “DisCoverage for Non-Convex Environments with Arbitrary Obstacles”, In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4486–4491, Shanghai, China, May 2011.
- A. Breitenmoser, J.-C. Metzger, R. Siegwart and D. Rus, “Distributed Coverage Control on Surfaces in 3D Space”, In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5569–5576, Taipei, Taiwan, Oct. 2010.
- G. Caprari, A. Breitenmoser, W. Fischer, C. Hürzeler, F. Tâche, R. Siegwart, P. Schoeneich, F. RoCHAT, F. Mondada and R. Moser, “Highly Compact Robots for Inspection of Power Plants”, In *Proc. of the 1st International Conference on Applied Robotics for the Power Industry (CARPI)*, pp. 1–6, Montreal, Canada, Oct. 2010.

- A. Breitenmoser, F. Tâche, G. Caprari, R. Siegwart and R. Moser, “MagneBike: Toward Multi Climbing Robots for Power Plant Inspection”, In *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1713–1720, Toronto, Canada, May 2010.
- A. Breitenmoser, M. Schwager, J. Metzger, R. Siegwart and D. Rus, “Voronoi Coverage of Non-Convex Environments with a Group of Networked Robots”, In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4982–4989, Anchorage, USA, May 2010.
- M. Rossi, C. Lombriser, A. Breitenmoser and G. Tröster, “Smart Dice”, In *Adjunct Proc. of the 3rd European Conference on Smart Sensing and Context (EuroSSC)*, Zurich, Switzerland, Oct. 2008.

Workshops (refereed)

- J. Alonso-Mora, A. Breitenmoser, S. Wismer, R. Siegwart, P. Beardsley, “Human-Robot Shared Control in a Large Robot Swarm”, Poster Presentation, ICRA Workshop on *Many-Robot Systems: Crossing the Reality Gap*, IEEE International Conference on Robotics and Automation (ICRA), St. Paul, USA, May 2012.
- C. Lombriser, A. Bulling, A. Breitenmoser and G. Tröster, “Speech as a Feedback Modality for Smart Objects”, In *Proc. of the 2nd International Workshop on Intelligent Pervasive Devices (PerDev)*, Galveston, USA, March 2009.

Conference Videos and Demonstrations (refereed)

- J. Alonso-Mora, A. Breitenmoser, M. Ruffli, S. Haag, G. Caprari, R. Siegwart and P. Beardsley, “DisplaySwarm: A Robot Swarm Displaying Images”, *Symposium: Robot Demonstrations*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, USA, Sept. 2011.

Technical Reports

- C. Lombriser, M. Rossi, A. Breitenmoser, D. Roggen and G. Tröster, “Recognizing Context for Pervasive Applications with the Titan Framework”, *Technical Report*, Wearable Computing Laboratory, ETH Zurich, Zurich, Switzerland, 2009.

Theses

- A. Breitenmoser, “Titanic Smart Objects”, *Master’s Thesis*, Eidgenössische Technische Hochschule (ETH), Zurich, Switzerland, April 2008.
- A. Breitenmoser, “Simulation and Measurement of Silicon Carbide Resonators for Harsh Environment”, *Bachelor’s Thesis*, University of Edinburgh, Edinburgh, Scotland, and Eidgenössische Technische Hochschule (ETH), Zurich, Switzerland, June 2006.

Patents and Patent Applications

- P. Beardsley, J. Alonso-Mora, A. Breitenmoser, F. Perazzi, A. Hornung, “Robotic Texture”, Disney Enterprises, 2012: *US 20120206473*.
- P. Beardsley, J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, I. Matthews and K. Yamane, “Display with Robotics Pixels”, Disney Enterprises, 2011: *US 20110304633*.
- M. G. Zahnert, E. Fonseca, A. Ilic and A. Breitenmoser, “Automatic Sizing of Images Acquired by a Handheld Scanner”, Dacuda AG, 2010: *US 20100296129*.
- M. G. Zahnert, E. Fonseca, A. Ilic and A. Breitenmoser, “Mode Switching in a Handheld Scanner”, Dacuda AG, 2010: *US 20100296133*.
- M. G. Zahnert, E. Fonseca, A. Ilic, S. Meier and A. Breitenmoser, “Image Processing for Handheld Scanner”, Dacuda AG, 2010: *US 20100295868*.
- M. G. Zahnert, E. Fonseca, A. Ilic, S. Meier and A. Breitenmoser, “Real-time Display of Images Acquired by a Handheld Scanner”, Dacuda AG, 2010: *US 20100296131*.

Academic Service

Reviewer for Journal

- Transactions on Robotics
- The International Journal of Robotics Research
- Mechatronics
- Neural Computing and Applications

Reviewer for Conference

- International Conference on Autonomous Agents and Multiagent Systems (AAMAS)
- International Conference on Applied Robotics for the Power Industry (CARPI)
- International Symposium on Distributed Robotic Systems (DARS)
- International Conference on Robotics and Automation (ICRA)
- International Conference on Intelligent Robots and Systems (IROS)
- International Workshop on the Algorithmic Foundations of Robotics (WAFR)