VĚDECKÉ SPISY VYSOKÉHO UČENÍ TECHNICKÉHO V BRNĚ

Edice PhD Thesis, sv. 442

ISSN 1213-4198

Ing. Mgr. Petr Švec

Using Methods of Computational Geometry in Robotics

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

Ing. Mgr. Petr Švec

USING METHODS OF COMPUTATIONAL GEOMETRY IN ROBOTICS

VYUŽITÍ POČÍTAČOVÉ GEOMETRIE V ROBOTICE

Short version of Ph.D. Thesis

Study field:	Konstrukční a procesní inženýrství Design and Process Engineering
Supervisor:	Doc. RNDr. Ing. Miloš Šeda, Ph.D.
Opponents:	Prof. Dr. RNDr. Lubomír Smutný Prof. Ing. Vladimír Řeřucha, CSc. Doc. Ing. Ivan Zelinka, Ph.D.
Presentation date:	18. 12. 2007

Keywords: autonomous robot, Voronoi diagram, Generalised Voronoi Diagram, Fortune's plane sweep algorithm, robot motion planning, robot localisation, map generation, group robot motion planning, Reynold's flocking algorithm

Klíčová slova: autonomní robot, Voroného diagram, zobecněný Voroného diagram, Fortuneho zametací algoritmus, plánování pohybu robota, lokalizace robota, generování mapy, plánování pohybu skupiny robotů, Reynoldův houfující algoritmus

Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology

© Petr Švec, 2008
ISBN 978-80-214-3570-4
ISSN 1213-4198

CONTENTS

1 INTRODUCTION AND THESIS OBJECTIVES

The thesis deals with the motion planning of an *autonomous robot*[1] in an unknown or partially known indoor or outdoor environment [37]. The solved tasks include the *global* and *real-time motion planning*[2] between two given positions[3] for a polygonal robot, which has the largest possible (the safest) distance from surrounding obstacles (see Figure 1.1), where uncertainty, localisation, kinodynamic properties of the robot (see Figure 1.2), and path relaxation techniques[4] are also considered. Furthermore, a map generation and exploration task is solved together with the motion planning for groups of robots and their formations.

In the thesis, the methods presented solve these tasks efficiently by means of the *computational geometry* [13]. The computational geometry emerged from the field of algorithms design and analysis in the late 1970s. Its success of the problems studied, practical and efficient solutions from the asymptotic time complexity point of view, and its huge range of application domains laid grounds for its future expansion into robotics.

One of the most useful structure in the computational geometry for the robot motion planning is the *Voronoi diagram* [13, 28]. It has a lot of applications in various fields since it preserves the largest distance from surrounding point generators. This property can be employed as a base for several motion planning tasks, where these generators are formed by obstacles. In addition to this, its *retraction property* assures that the robot is always capable of transferring itself onto this diagram along a collisionless sight line, from which follows, that the Voronoi diagram entirely captures the continuity of the whole space as a topological graph [28]. The extension of this diagram for point, segment, or polygonal generators is called the *generalised Voronoi diagram* [28].

The usability of this generalised diagram for the robot motion planning is conditioned by an existence of efficient, robust, and practical algorithms for its computation. There are presented two designed algorithms in the thesis, which are accompanied by their asymptotic time complexity analysis. The added value of these approximation algorithms lies in robustness and simplicity of their implementation and high computational efficiency in comparison to algorithms of the same class.

---

[1] "Autonomous robots are robots which can survive in unstructured environments without continuous human guidance. Unlike factory robots they must negotiate environments which are changeable, full of obstacles and eventually hostile." (Webster's Online Dictionary).

[2] The robot does not have *a priori* knowledge of the environment.

[3] An important part of the robotics is to have an algorithm capable of processing a high-level task of moving a robot from an initial position to a final one. A classical version of this problem is referred to as the *Piano's Mover's Problem*.

[4] In the case when the path does not have to strictly preserve the requirement of being in the largest possible distance from surrounding obstacles.

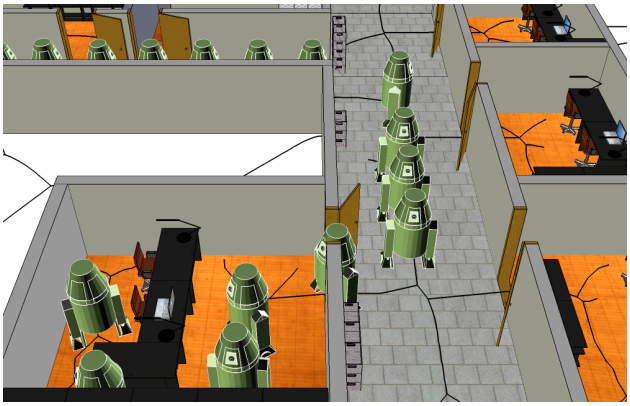

Figure 1.1: Robot motion planning in an office-like environment based on the generalised Voronoi diagram.

The robot motion planning is an enormous and exciting field. It originally ignored dynamics and other differential constraints, uncertainties, processing errors, optimality, and further extensions used nowadays. It has a lot of flavours and countless applications. These include hazardous duty services like demining military zones, cleaning toxic waste, repairing nuclear power plants, autonomous safe driving, parking-assist systems (parking cars and trailers, for example airport baggage trailers), cleaning windows on high buildings, aerospace applications (autonomous flying through the air or in space – designing safe trajectories), or maze exploration.

The perspectives of the robot motion planning are biped android walkers which can co-exist in human society, robots helping handicap people, or nanotech medical bots (small microscopic robots in patient's bloodstream).

Thesis Objectives The thesis addresses different types of motion planning tasks based on the generalised Voronoi diagram for an autonomous robot in an unknown or partially known environment. The primary objective is to design an algorithm for computing the generalised Voronoi diagram efficiently for a complex environment with point, segment, or polygonal obstacles (or polyhedral obstacles as the 2D representation of the environment is given by the robot's sensors) and to plan a motion of the robot on the shortest path between two given positions in this diagram.

In short, the objectives include:

- Design of a new approximation geometric algorithm for computing the generalised Voronoi diagram, which represents a trade-off between the computational efficiency, robustness, and implementation difficulty while preserving the sufficient precision for most applications like the robot motion planning. The algorithm should be accompanied by a detailed theoretical discussion including the proof of its asymptotic time complexity. Also a

comparison to existing techniques reflecting the current state of literature should be accomplished.

- Global motion planning (or multi-query motion planning) for a robot between two given positions. This presents a problem of finding a path for a robot, which has the largest (the safest) distance from surrounding obstacles in a priori known static environment.
- Real-time motion planning (or single-query motion planning) for a robot between two given positions. This presents a problem of finding a path for a robot, which has the largest (the safest) distance from surrounding obstacles in an unknown environment.
- Improvement of the found shortest path in the generalised Voronoi diagram in such a way that conforms to the robot's kinodynamic properties.
- Solution of the map generation and exploration task for the robot to be able to autonomously explore and map the whole a priori unknown environment based on the generalised Voronoi diagram and other computational geometry algorithms.
- Group robot motion planning exploiting the main substantial properties of the generalised Voronoi diagram.
- Analysis and discussions about the robot's localisation techniques and possible ways of improving it by the generalised Voronoi diagram, which consequently leads to improving uncertainty.
- Carrying out experiments using an implemented motion planning simulator for proving the theoretical discussions in practice.

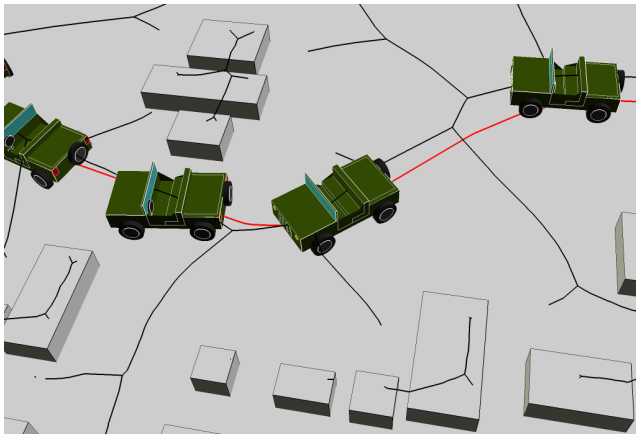

Figure 1.2: Kinodynamic planning of a car-like robot based on the generalised Voronoi diagram.

2 THEORETICAL BACKGROUND AND PREVIOUS WORK

This chapter presents the necessary theoretical background and references to previous work, which are closely related to the main results provided in the next chapter.

2.1 VORONOI DIAGRAM

The Voronoi diagram (see [13, 28], further VD) presents a way of dividing a m-dimensional continuous space into a set of regions, where all locations in that space are associated with the closest isolated points (so-called generators). Its counterpart is the Delaunay triangulation (see [13, 28], further DT), see Figure 2.1, where VD is marked by a full line and DT is marked by a dashed line.

A major reason for persisting success of Voronoi diagrams is that it can be generalised in a diversity of ways including 3D and higher dimensional varieties. There are a lot of variants (different types of generators and distance measures) of this diagram, which are covered in [16, 28].

Given a finite set $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, where $1 \leq n < \infty$, of n point generators in the plane. Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ be location vectors of p_i and p_j. Then, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ for $i \neq j, i, j \in \mathbb{N}_n$ ($\mathbb{N}_n$ is a set of native numbers with the size n). All sites in the space are assigned to their nearest point generators from P with regard to the Euclidean distance. The result is a transformation of $\mathbb{R}^2$ into a set of regions

$$V(p_i) = \{\boldsymbol{x} | \parallel \boldsymbol{x} - \boldsymbol{x}_i \parallel \leq \parallel \boldsymbol{x} - \boldsymbol{x}_j \parallel \text{ for } \forall \boldsymbol{x} \in \mathbb{R}^2, i, j \in \mathbb{N}_n; j \neq i\} \qquad (2.1)$$

associated with generator p_i. The *ordinary Voronoi diagram* (further VD) is given by a set of regions $VD(P) = \{V(p_1), \ldots, V(p_n)\}$ generated by a set of

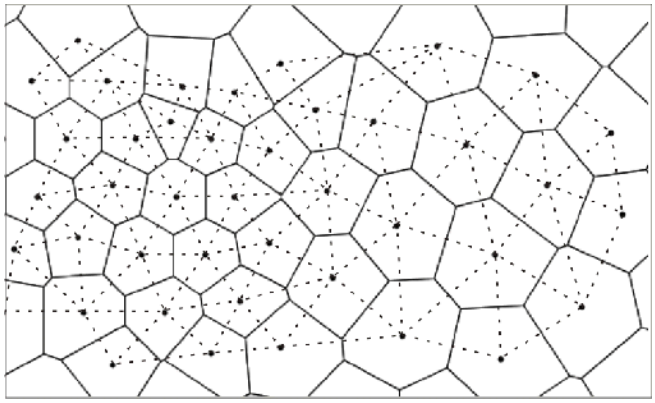

Figure 2.1: Voronoi diagram (full line) for point generators together with the Delaunay triangulation (dashed line), which makes its counterpart.

generators P. All points laying on the *VD* are assured to have the largest distance from surrounding point generators.

Generalised Voronoi diagram Given a set $O = \{o_1, \ldots, o_n\} \subseteq \mathbb{R}^2 (1 \leq n < \infty)$ of n generators in the plane, their *generalised Voronoi diagram* [28] (further GVD) is a partition of the plane into regions, one for each generator, such that the region of generator $o_i \in O$ contains all locations of the plane that are closer to o_i than to any other generator $o_j \in O$. The generators in O can be a set of points, segments, polygons, areas, polyhedrons, etc.

2.1.1 Voronoi Diagram Computation Overview

To evaluate an efficiency of an algorithm that carries out a computation over a set of input data, a standard approach, that is not susceptible to implementation or hardware, is needed. This approach evaluates the asymptotic behaviour of the time required by the algorithm with respect to the size of input data [28].

An overview, including time complexity of basic algorithms, for a deterministic construction of the Voronoi diagram with varying metrics can be found in [16].

By using efficient techniques and data structures, the time complexity of the Voronoi diagram computation can equal to $O(n \log n)$. This is the case of the *Fortune plane sweep algorithm* (see [13, 15, 28, 37]), which is based on the fundamental *plane sweep technique* of the computational geometry.

The main idea of this algorithm is a shift of a horizontal line l (called a sweep line) from the top to the bottom part of a plane over all n generators $p \in P$, where $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^2 (1 \leq n < \infty)$, while constructing the $VD(P)$. During this shift, a sequence of parabolic arcs (so-called a beach line) is maintained consisting of points, which have the same distance from the sweep line l and the generators of these parabolic arcs, as is illustrated in Figure 2.2.

The proof of the plane sweep algorithm can be found in [13] and [28], whereas some practical ideas for improving its implementation and making it more robust can be found in [26, 37].

During the computation, the breakpoints between these parabolic arcs in the beach line trace the $VD(P)$, and the part of the plane above l does not affect the computation in comparison to the part of the plane below l. A change in the structure of the beach line is affected by two basic events – a creation of a new parabolic arc producing a new *site event* or shrinking of an existed parabolic arc producing a new *circle event*.

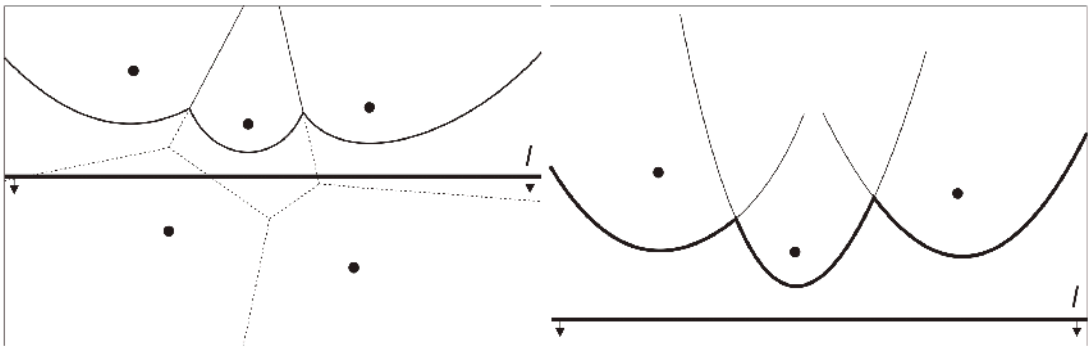

Figure 2.2: Beach line – a sequence of parabolic arcs.

Refer to [37] or [36] for a detailed description of this algorithm including description of its implementation and handling degenerated cases.

2.1.2 Generalised Voronoi Diagram Computation Overview

The *incremental algorithm* for *GVD* computes this diagram for a set of point generators in advance, and then segment generators without their end vertices are being added one by one, thereby modifying the diagram [28]. The implementation difficulty of this algorithm lies primarily in direct handling parabolic arcs.

The *divide and conquer algorithm* for computing *GVD* recursively approximately divides a set of generators into two sets until all subsets contain only two generators [28]. In the next step, these sets are being connected into larger units, until final *GVD* is created. The complexity of this algorithm is situated in unifying sets and generating symmetry axes (edges of GVD).

The Fortune plane sweep algorithm serves also as a base in the *VVP* technique [25], which uniformly approximates segment generators by point generators and combines the Voronoi diagram, the *Visibility graph* [13, 28], and the *Potential field method* [13] into one technique for motion planning.

An approximation algorithm for constructing *medial axis* (a set of centers of circles, which touch the inner surface at more than one point) in 2D or 3D environment is presented in [3]. This algorithm adapts the density of point generators in particular places. The main idea is to efficiently generate a small set of partially overlapping maximal spheres to enclose the entire space. These spheres are then further approximated in order to identify points that are in proximity to the medial axis.

The generalised Voronoi diagram can also be computed from digital images of generalised polygons, see [31] or [21]. Graphics rasterization hardware is used for computing the *GVD* in [18]. Also in [12] the authors present an approximation algorithm based on graphics hardware. Further theory behind the computation of the *GVD* based on a raster is presented in [28]. The obvious

disadvantage of this approach is a need for a mobile robot to be equipped with a powerful graphics hardware.

Some solutions for motion planning of robots with many degrees of freedom [19] are based on the *hiearchical generalised Voronoi diagram* (further HGVD) defined in detail in [4, 5, 7]. The approximation incremental construction of the *HGVD*, originally developed for sensor based motion planning, is described in papers [6, 9] and further improved in [8, 11], other modifications and applications can be found in [10].

2.2 ROBOT MOTION PLANNING

Motion planning is one of the most challenging task in robotics. An autonomous robot knows where it is, it knows where it is going but it does not know how to get to the required position. The motion planning is a process, that transforms a semantic description to a sequence of numerical descriptions of motion. It relies on success of the following parts of robotics: perception, localisation, cognition, and motion control.

The purpose of the motion planning is to determine *feasible paths*[1] or *trajectories*[2] in an unknown or partially unknown environment.

There are two basic criteria imposed on the constructed path. The first is the $feasibility$ criteria, where the constructed path has to allow the robot to get to the goal position, regardless of the path efficiency. The second one is the *optimality* criteria, where the path is optimal in some specific manner (length, time). For most problems, feasibility is already challenging enough, therefore achieving optimality is considerably harder.

2.2.1 Basic Motion Planning Methods

Motion planning methods can be divided into two classes – the *single-query* (or real-time, it assumes to have only single initial and final configurations as an input) and the *multi-query* (or global, it involves numerous queries) motion planning methods.

In the multi-query version, it is suitable to preprocess the whole environment so that the future queries can be answered efficiently. Therefore this version of motion planning includes two main steps [23]:

- Preprocessing – the result is a representation of the free configuration space C_{free} (set of all possible robot's configurations or placements, see

[1] A feasible path is a collision-free path guaranteeing the effective execution of a specified actions especially in presence of kinodynamic models. It must satisfy some constraints imposed by the workspace or by the robot itself.

[2] A trajectory is a path with dynamic constraints like velocity and acceleration.

[37]), among all obstacles in a given area, using a graph or function.

- Query processing – searching graph or using a function to find a path.

The motion planning method depends on the description of the robot's environment representation. This can be a continuous geometric description, a decomposition-based geometric map, or even a topological map. A detailed comparison of the cell decomposition and roadmap methods can be found in [32].

The three basic classes of methods for robot motion planning are as follows:

- Cell Decomposition – a decomposition of C_{free} into cells. The continuity representation of this decomposition is made by using a neighbourhood graph of cells (introduced by Latombe [23]).
- Roadmap – a continuity representation of C_{free} by a graph.
- Potential Field – a mathematical function is defined over C_{free}, which has a global minimum in the goal configuration and maximum in places of given obstacles.

The *retraction approach* belongs to a family of methods for constructing roadmaps [23]. Therefore it can be used for a direct construction of a graph representing C_{free} continuity and the problem of finding a path amid obstacles $CB \in C_{obs}$ is reduced to a problem of finding the path in a topological graph. The important property is that the generated graph has the largest distance from surrounding obstacles (generators).

2.2.2 Motion Planning with Kinematic and Dynamic Constraints

The robot's free configuration space C_{free} defines the range of possible configurations, which the robot can achieve in its environment. All the robot's possible configurations are defined by its controllability. This controllability is given by the robot's kinematic model, which is given by the robot's underlying mechanism. This kinematic model, to some extent, limits the robot's ability to move in a certain way. The robot dynamics is related to the robot's inertia and adds another additional constraints on C_{free} and trajectory due to mass and force considerations.

A robot must be able to correctly move between two configurations on the constructed path. The nonholonomic constraint imposes that the curvature of a resulting path should have a maximum limit. For a robot in order to stay on the trajectory, it must exert effort to overcome the centrifugal acceleration, therefore a circular arc on the path should have a minimum curvature radius.

3 MAIN RESULTS

Challenging implementation of the plane sweep algorithm [15] and other methods for computing the *GVD* for segment generators leads further research to approximation algorithms, that represent a trade-off between speed of computation and implementation difficulty, and are easy made to be numerically robust. For most applications (like the robot motion planning) the computation of an approximated Voronoi diagram within a given precision is sufficient.

The presented *basic uniform approximation algorithm* uniformly approximates segment or polygonal generators by point generators first (see [28]) and then applies the modified Fortune plane sweep technique (see [37]) to compute the *GVD* over this set. Due to the uniform approximation nature, this algorithm is very slow, thus two new approximation algorithms, which attempt to put approximation points only to places where needed, have been proposed.

The following sections of this chapter deal with the robot motion planning based on the *GVD* including the robot's kinodynamic constraints, the map generation and exploration task, and finally with the motion planning of a group of robots.

3.1 GENERALISED VORONOI DIAGRAM COMPUTATION

3.1.1 Non-Uniform Real-Time Approximation Algorithm

A new approximation algorithm for constructing the *GVD* for point, segment, or polygonal generators has been developed as introduced in [35]. This algorithm is based on the Fortune plane sweep technique, which combines advantages of being optimal like the divide-and-conquer algorithm but avoiding the difficult merge step, and being relatively simple like the incremental algorithm, while representing a trade-off between a complexity of implementation and a speed of computation.

The main idea of this algorithm is a non-uniform approximation of every segment generator (or series of segment generators) by sequences of point generators with higher density in narrow corridors (see Figure 3.1). This approach attempts to efficiently detect edges of narrow corridors (segment generators), which are approximated by more point generators than others, thereby the computation is faster in comparison to uniform point distribution with the same precision for all generators.

This approximation process can be divided into two parts. The first part is responsible for an initial uniform coarse approximation with a given precision K, which defines a length of subsegments laying between approximation point generators on segment generators or segments of polygonal generators. This initial uniform approximation is optional and it serves for acquiring bet-

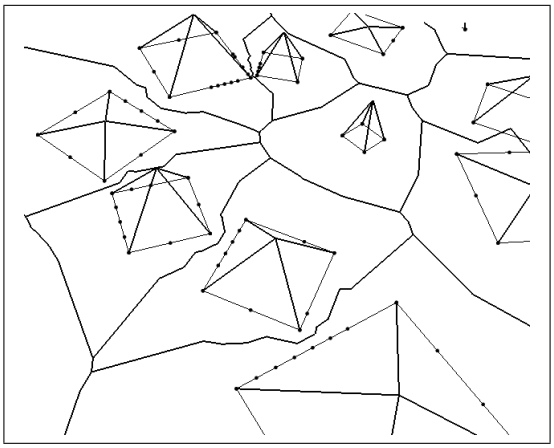

Figure 3.1: Generalised Voronoi diagram for a 3D environment with spikes computed using the real-time non-uniform approximation algorithm.

ter precision of the diagram in the second part of the approximation process. The second part is more complex and the approximation process is applied only when two neighbouring generators are close enough. The logic responsible for the approximation process itself is directly put into the underlying implementation of the Fortune plane sweep technique, refer to [37].

Theorem 1. *The asymptotic time complexity of the real-time non-uniform approximation algorithm for computing the GVD is* $O(n \log n)$ *for* n *input segment generators (including segments of polygonal generators), where the resolution of approximation point generators on each segment generator is upper bounded by a precision constant* K*.*

See [37] for a proof of this theorem.

3.1.2 Approximation Algorithm With Fast Preprocessing

A new approximation algorithm with fast preprocessing for constructing the *GVD* of segment or polygonal generators has been developed. The algorithm is based on the already presented idea, where segment generators, based on their proximity, are approximated by a sequence of point generators, thereby detecting narrow corridors in the space.

This new approach is focused on two stages, the first – segment or polygonal generators fast preprocessing stage and the second – computing *GVD* using the Fortune plane sweep algorithm stage itself. The already mentioned – real-time algorithm – has no need for the preprocessing part, however this comes in price regarding the resulting precision of the *GVD*.

The algorithm is divided into two parts. The first part is responsible for a fast uniform preprocessing of segment or polygonal generators (the non-uniform version can be considered as well). This preprocessing includes a

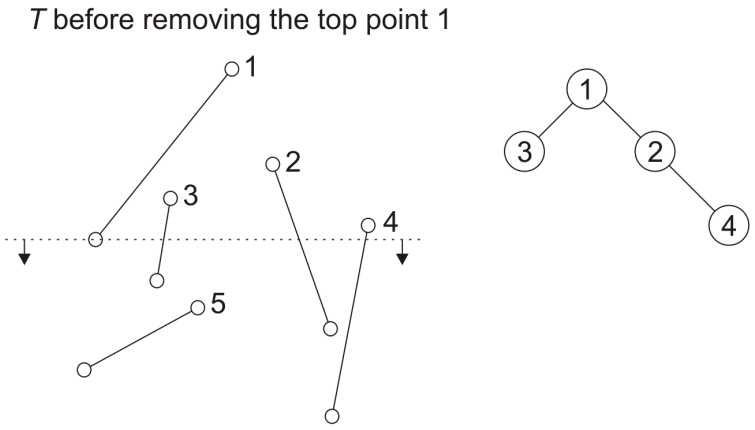

Figure 3.2: The status structure T.

point approximation process, that is applied only when two neighbouring generators are close enough. It is based on the plane sweep technique to be able to detect the closest neighbouring segment generators in $O(n \log n)$ time complexity, where n is a number of input segment generators. Having applied the neighbouring detection procedure, these segments are uniformly updated by new approximation point generators.

The second part is made by the modified Fortune plane sweep algorithm as described in the thesis. Having constructed the GVD for approximation point generators made in the preprocessing part, the redundant edges are deleted or not considered during further computation on this structure.

The main data structures of this algorithm include the priority queue Q, the status structure T, and the visibility structure V.

Status Structure The status structure T is represented as AVL tree [13, 22]. It serves for an efficient detection of neighbouring horizontal segments in $O(n \log n)$ time complexity. It stores only top end points of segments detected on the plane sweep line (see Figure 3.2).

Visibility Structure The visibility structure serves for an efficient detection of neighbouring vertical segments in $O(\log n)$ time complexity and is represented as AVL tree. This structure belongs to the category of dimensional range searching data structures [13]. The input data is a set of segments $S \leftarrow \{s_1, s_2, \ldots, s_n\}$ in one dimensional space. A query asks for a segment containing a given point or returns segments inside a one dimensional interval $[s : s']$. The main operations are adding a new segment, and finding and removing a set of segments within an interval. Each node in the tree stores a visibility segment, which points to its original input segment, to guide the search.

Algorithm details The algorithm starts by storing end points of all segment generators into the priority queue Q, where these end points are represented as events and sorted according to their y coordinates. During the process, these events are being dequeued from Q and handled according to their type.

If an event e_p represents the top point of its parent segment generator, neighbouring horizontal segment generators of this point are found in the status structure T and their point resolution is changed.

To detect neighbouring horizontal segment generators using the status structure T, traverse through this tree of top segment end points. The algorithm starts in the root node of T and descends into its left or right child node according to the x coordinate of the event. During this descend, it maintains current left and right neighbouring segment generators. Once a leaf node is detected, the latest detected left and right neighbouring segment generators given by their end points are returned.

The point resolution of a given segment generator can be both, the uniform or the non-uniform point distribution. In case of using the uniform point distribution, the distance between points on a segment generator represents the value of resolution, and is given by the precision approximation factor P.

The next step is inserting the top point of the segment generator into the status structure T. This is followed by changing of resolution of segment generators immediately above currently processed segment generator. These segment generators are detected using the visibility structure V as described earlier. During this process, a few visibility segments disappear and maximum three new visibility segments in V are created.

If the event e_p represents a bottom point of its parent segment generator, the top point of this generator is removed from the status structure T. This is followed by the horizontal resolution update procedure.

Theorem 2. *The asymptotic time complexity of the preprocessing part of the approximation algorithm with fast preprocessing is $O(n \log n)$ for n input segment generators (including segments of polygonal generators).*

Theorem 3. *The asymptotic time complexity of the approximation algorithm with fast preprocessing is $O(n \log n)$ for n input segment generators, where the resolution of approximation point generators on each segment generator is upper bounded by the precision approximation factor P.*

The formal descriptions of both algorithms and proofs of their time complexities can be found in the thesis [37].

3.1.3 Analysis of Experiments

A few experiments have been carried out to prove main advantages of the real-time non-uniform approximation algorithm and the approximation algo-

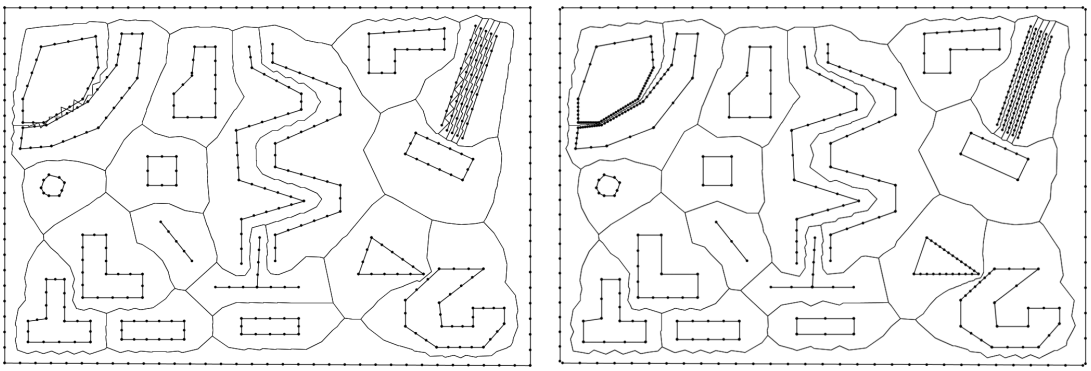

Figure 3.3: Result of the computation of the basic uniform approximation algorithm (left) and the result of the computation of the approximation algorithm with fast preprocessing (right).

rithm with fast preprocessing over the basic uniform approximation algorithm (for an example, see Figure 3.3). Moreover, there is drawn a comparison between them. These algorithms were implemented in the Java programming language and they run on AMD Sempron 3000+ machine.

According to the accomplished experiments, there is a clear evidence, that the real-time non-uniform approximation algorithm and the approximation algorithm with fast preprocessing are substantially faster than the basic uniform approximation algorithm, especially for large and detailed environments. The main reason for this is the amount of approximation point generators, which are put in higher density only into problematic places like narrow corridors to achieve sufficient precision. Not only the amount of approximation point generators is important but also their distribution. Based on these experiments, the less uniform distribution, the better efficiency of the Fortune plane sweep algorithm is.

Even though the environment is very dense, the speed of the preprocessing part of the approximation algorithm with fast preprocessing is very high. The percentage from the whole time spent on computing of this algorithm over a small amount of generators is actually dropping to insignificant part as the amount of generators is raising.

In the case of the basic uniform approximation algorithm, the sufficient precision of approximation is not known in advance as in the case of the approximation algorithm with fast preprocessing (or real-time non-uniform approximation algorithm). This makes a substantial practical difference between these two algorithms since the sufficient precision for the basic one must be guessed.

Finally, according to experiments, the approximation algorithm with fast preprocessing is faster and more precise in particular situations than the real-time one hence it should be preferred.

3.2 ROBOT MOTION PLANNING

In the thesis, the *GVD* is used as a roadmap in the workspace for a robot to be able to find a path between two given locations while preserving the largest (the safest) distance from surrounding obstacles. The requirement of having the path in the largest distance from surrounding obstacles is not always practical and can be relaxed.

The latest research is focused on dealing with the workspace geometry instead of the geometry of the configuration space due to its very difficult and rather theoretical explicit construction for three dimensional or higher spaces[1]. The *GVD* can also serve as a base for sampling-based robot motion planning methods for highly articulated robots.

From the implementation point of view, it is almost impossible to construct a path preserving the optimality criteria. Due to this reason, approximation algorithms are used, that are capable of constructing a path, which is only an approximation to the optimal solution.

There are two strategies of solving the motion planning problem, the multi-query (or global) and the single-query (or real-time) ones. Both strategies have been implemented and results from the simulator follow.

3.2.1 Global Robot Motion Planning

In cases when a map of the whole environment is known and multiple motion planning queries are needed, the multi-query motion planning strategy is an optimal solution. In this case a sequence of the following steps is applied:

1. Computation of the *GVD* for a given environment by using either the non-uniform real-time approximation algorithm or the approximation algorithm with fast preprocessing.

2. Computation of a feasible path from an initial to a goal configuration for a robot using a graph search algorithm like the $A*$ algorithm.

3. An optional application of a path smoothing technique [38].

Two experiments of the multi-query motion planning in indoor and outdoor environments have been carried out. They show *GVD* based on a set of approximation points, which are well distributed in narrow corridors, thereby preserving the sufficient precision for the motion planning task. In this task, the robot is moving along the shortest smooth collisionless path.

Each vertex in the Voronoi diagram has the largest distance from surrounding generators. The Fortune plane sweep algorithm can record this information

[1] In cases when the robot has three or more degrees of freedom.

into all vertices of the DCEL. This clearance information can be used for finding a path for robots with different shapes and sizes and also for robots with different velocities and accelerations, thus kinodynamic constraints.

3.2.2 Real-Time Robot Motion Planning

In cases when a map is unknown or partially known, only single motion planning query is needed, the single-query motion planning strategy is an optimal solution. In this case a sequence of the following steps is applied:

1. Utilisation of a sensor to get a suitable representation of local environment (high-level features).

2. Computation of the GVD for the local environment by using either the real-time non-uniform approximation algorithm or the approximation algorithm with fast preprocessing.

3. Computation of the local feasible path for a robot using a graph search algorithm like the $A*$ algorithm.

4. An optional application of a path smoothing technique.

5. Connection of the local GVD_{new} to the general GVD using the algorithm GVDConnect (see [37]) and return to the step 1 until the goal is reached. The GVD_{new} is connected to the GVD only if the GVD_{new} contains parts, which are not presented in the original GVD, and the distance between the GVD_{new} and GVD is larger than a given constant (for not connecting the very similar GVDs). Thus, sequences of small-scale local maps and GVD trees are generated, that form the final map and GVD.

Two experiments of the single-query motion planning have been carried out. In the first experiment, the robot does not have a map of the environment, it only knows its initial and goal positions. The final computed path can be reused later by other robots to simplify their navigation through the environment.

The second experiment proves the ability of the robot to follow a path based on the GVD to handle a cyclic environment correctly and to confirm its ability to improve its own localisation by matching GVD subtrees independently on its position [1].

3.2.3 Motion Planning with Kinematic and Dynamic Constraints

The task of the *planner*[2] is to compute a trajectory through a *state space*[3] that connects initial and goal states while satisfying the robot's differential constraints. The planner has to consider its kinematic or dynamic constraints by either postprocessing the path (smoothing) or directly including the kinematic and dynamic models into the computation.

The objective of the kinematic controller is to follow the path made up of a sequence of key vertices and segments. An example of an applied car-like kinematic model on generated path laying on the *GVD* is shown in Figure 1.2. In this example, during the computation, the key points on the found path are determined first, and then the path between these key points is interpolated with respect to a chosen kinematic or dynamic model. The differential constraints are ignored in the planning process (construction of the GVD) first to be appropriately handled later on. This corresponds to executing the computed path as closely as possible using control techniques. A better approach could be considering differential constraints directly in the planning process, which conforms to the natural motions of mechanical system.

The corridor made along the path consists of maximal clearance circles of Voronoi vertices. This corridor can be used for constructing an arbitrary path, which suits a particular application regardless of the shape of the original path. The Figure 3.4 shows this situation, where the original path strictly lays onto the *GVD* while the other path is short enough and preserves some amount of clearance between the robot and obstacles. This path can be computed in $O(n)$ time complexity, where n is a number of path vertices.

3.3 MAP GENERATION AND EXPLORATION

In some tasks like rescue operations, a robot must be able to autonomously scour the whole environment (house, ruins, cave, etc.) to find and save human beings or animals. These tasks involve all components of the robot motion planning – perception, localisation, cognition, and motion control. The robot in a map generation and exploration task should be able to autonomously explore the whole environment with its on-board sensors, extract knowledge from raw sensor data, interpret it by building an accurate map, and estimate its position relative to this map.

An algorithm for map generation and exploration based on the *GVD*, representing a topological map of the configuration space, has been developed under the assumption of having an accurate localisation technique.

[2]In the artificial intelligence notation, the planner stands for an algorithm responsible for constructing a path [40, 49, 54].

[3]It can be a configuration space or a phase space of this configuration space, see [24]

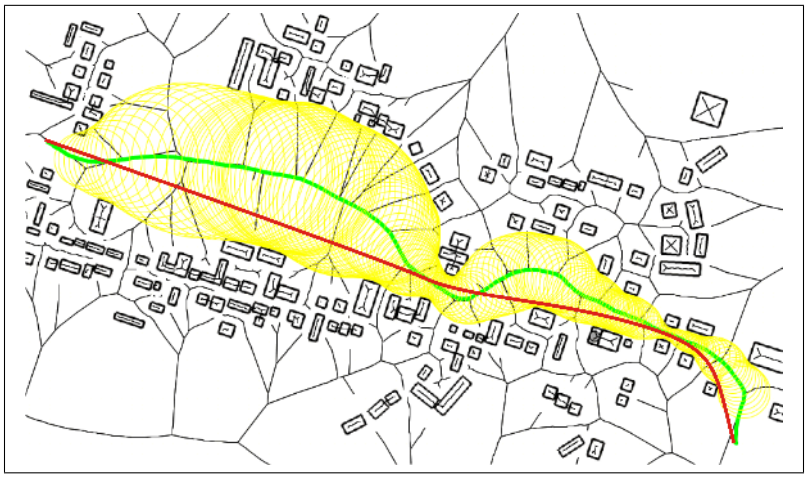

Figure 3.4: Corridor which is encoded into the path during the computation can be used to create different types of paths for different applications later on. There is shown a path strictly laying onto the generalised Voronoi diagram and the other one that is computed to be used in a relevant application.

Algorithm outline The robot starts in an initial position and generates a local map of its surrounding environment and initial *GVD*. Thereafter, according to the *wall-following technique* [2], *DFS* technique for graph searching [24], or so-called *recursive exploration* [2], it starts exploring this new generated graph. Having arrived to a vertex with a degree of edge incidence at least 2, it generates a new *GVD*, limit it to a given radius r, and attempts to connect it to the global *GVD*. This new GVD_{new} is connected to the global one only if it has parts, which are not presented in it, or the global *GVD* has a larger distance from GVD_{new} than is the defined distance δ. Thus, sequences of small-scale local maps and *GVD* trees are generated. This technique is also used for localisation by matching *GVD* subtrees independently of the robot's position, and is capable of handling cyclic environments correctly.

An experiment built on the map generation and exploration task can be found in the thesis [37]. The map shown is autonomously generated by a robot after it has been deployed into an unknown environment. The generated map can further be used for different types of localisation.

3.4 GROUP MOTION PLANNING

In the thesis, the group motion planning task deals with the collisionless motion planning of a large group of robots between two given positions, see Figure 3.5. This group of robots should keep its coherence while moving and the largest possible distance from surrounding obstacles, provided that it has been given a *lateral dispersal* (it is perpendicular to the path) and the *longitudinal* one (the distance between the most and the least advance positions of

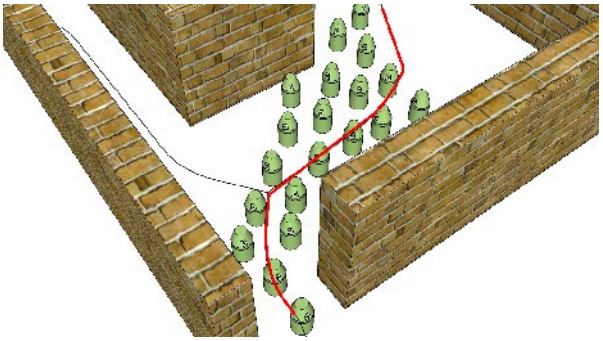

Figure 3.5: Group motion planning based on the generalised Voronoi diagram and Fortune's flocking rules.

robots on the path). Possible applications of this task are service operations, where each robot in the group carries different apparatus or can be equipped with different sensors.

In theory, the standard planning algorithm can be applied directly to this task for planning multiple robots in an unknown environment [24]. However, the dimension of the state space X would make the time of the computation exponential[4]. This approach of treating multiple robots as a one composite system belongs to the *centralised* [24] planning schemes. These have an advantage of finding a solution if one exists that is why they are called *complete*. Conversely, the *decoupled* planning scheme generates independent paths for all robots first and then resolves interactions between them. The *prioritised planning* is an example of this scheme [24].

The motion planning of multiple robots can be decoupled in many ways [24], however for the transportation task, which is the goal of the thesis, it is reasonable to think of the group as a single entity, where individuals of the group control themselves according to certain simple rules.

The motion planning for a group of robots can be driven by the *swarm theory* (or swarm intelligence field, [17, 20, 29]). The swarm theory is inspired by biology, especially by studying social insects colonies. Social insects provide surplus of examples indicating that locally sensed stimulus and reflexive behaviour can result in global behaviours [14]. Flocks of birds, swarms of bees, and schools of fish present complex spatial behaviour, that is made if each agent follows a few simple rules.

The robotic swarm is fully autonomous and hence not controlled centrally, therefore robots also can respond quickly to a dynamic environment since they do not have to wait for an instruction from a central controller. The swarm has an increased sensing precision, thus it can localise itself faster and more accurately provided that the robots in the swarm exchange information about their positions regularly. The higher reliability of the swarm increases a chance

[4]The dimension of X grows linearly with respect to the number of robots.

to complete a task and it can be achieved by redundancy of robots easily.

A robotic swarm moves in order to perform some tasks. When the swarm is moving, the challenge is to have all robots avoiding obstacles that appear in their paths. There is an assumption that each robot knows the exact positions and velocities of other robots in order to compute the swarm center and the swarm average velocity.

Each robot has its own set of simple behaviours (actions) and cooperation of them in a swarm can result in a more complex behaviour. Craig Reynolds introduced a *flocking model* [30] (or behaviour) for motion planning of a swarm of entities called boids. This model, presented in its original form, was used to simulate flocks of birds, fish, or other creatures predominantly for computer graphics purposes. However, a modified version can be used to simulate or drive swarms movement of units, squads, or air squadrons. The flocking model presents self-directing autonomously operating entities without a centralized control.

The results in this section have been partly presented in [33] and further expanded in [34]. The robots in the group are driven by simple rules of the flocking model. The A* algorithm is used to find the shortest path τ between two given positions. This backbone path is made up by the generalised Voronoi diagram computed either in advance or computed in real-time by a lead robot of the group. Every robot should be able to pass the path τ with such minimum width, that is equal to the diameter of the smallest circle circumscribing the largest robot from the group.

Each vertex of the GVD contains a maximum clearance information. This information can be used for deciding whether all robots from the group will be able to pass the path. This path is divided into subgoals, which equals to vertices of the GVD on the path. The sequence of all these upper bounded clearance circles make a collisionless corridor along the path. The transfer of the group along the path in this corridor consists of partial transfers to subgoals (or vertices on the path) until the goal is reached. This corridor can be readily employed for planning of a group of robots to control its maximal lateral dispersion in different vertices on the path. The larger clearance leads to higher coherence of the group, therefore motion planning on the GVD is in this sense optimal [27].

During the path following behaviour, the vertices on the path being followed are made by the Voronoi vertices, where for each of them the clearance information is encoded. Thus, the group is moving along the path while preserving its lateral and longitudinal dispersal. A robot in this group always chooses the most advanced key vertex v with a clearance c but the one that is in its reach inside the corridor. This vertex will cause the robot to move in a given direction.

4 CONCLUSIONS AND FUTURE WORK

Two novel approximation algorithms for constructing the generalised Voronoi diagram have been proposed. These new algorithms – the real-time non-uniform approximation algorithm and the approximation algorithm with fast preprocessing are based on the Fortune plane sweep technique. They attempt to detect narrow corridors in an environment that are consequently sampled with a higher density of approximation point generators. This results in a higher computing speed of them in comparison to the speed of computation of the basic uniform algorithm, which uses a uniform point approximation on every segment generator with the same precision. The result of the approximation is that the parabolic arcs of the generalised Voronoi diagram are substituted by sequences of straight segments.

These new algorithms directly fall into the family of geometric algorithms and they represent a trade-off between the speed of computation, robustness, and implementation difficulty while preserving the sufficient precision for most applications. The time complexity of these algorithms is $O(n \log n)$ for n input segment generators (or segments of polygonal generators) provided that the resolution of approximation point generators on every segment is upper bounded. This theorem has been proved theoretically for both algorithms.

A few experiments on these algorithms have been carried out to give a reader an idea about their speed on current machines besides giving their asymptotic time complexities and also to compare their average speed to the speed of the basic uniform algorithm. These new algorithms are substantially faster especially for large and detailed environments. The main reason for this is an intelligent distribution of approximation point generators in a higher density in narrow corridors while not approximating segment generators, which are far away from each other, but still preserving a sufficient precision of the whole diagram for most applications.

The distribution of point generators is important since the less uniform distribution, the better efficiency of the Fortune plane sweep algorithm is, as follows from the shown experiments. The amount of time taken by the preprocessing part of the uniform algorithm with fast preprocessing makes only a negligible percentage from the whole computing time. This percentage is actually dropping as the amount of generators is raising, and this can be further improved by changing the uniform distribution to the non-uniform one, however, it can put an extra burden on the computation.

To show the practical aspects of these algorithms, there have been presented a few applications of them in the robot motion planning context, where the main task is a collisionless transfer of a polygonal robot amid point, segment, or polygonal obstacles from an initial position to a final one. Two main types of the motion planning task based on the generalised Voronoi diagram have

been solved – the global motion planning (or multi-query motion planning) and the real-time motion planning (or single-query motion planning).

In the real-time motion planning task the robot must be able to detect surrounding obstacles, represent the raw sensor data as high-level features (segments), compute the generalised Voronoi diagram, and proceed a series of local motion planning tasks to achieve the final position by having only a direction to it or by having only its local referenced coordinates. It has also been shown that the generalised Voronoi diagram can be used for improving the robot's localisation, which is a difficult subject in robotics.

The kinodynamics properties of the robot are an important part of the robot motion planning. These have been considered as well in both types of motion planning and shown in experiments. It is obvious that the rigidity of the generalised Voronoi diagram to preserve the largest or the safest distance from surrounding obstacles can be relaxed to accommodate the practicality of various motion planning tasks.

A new algorithm for the map generation and exploration task has been proposed under the assumption that the robot has precise localisation ability. This algorithm exploits the property of the generalised Voronoi algorithm to preserve the largest distance from surrounding obstacles, which suit the robot's sensors to capture the surroundings from a vantage point. In the exploration task the robot must be capable of exploring the whole previously unknown environment, where building its map is its indispensable part. Further, high level features from the raw sensor data must be extracted, interpreted as a map, and used to estimate the robot's position relative to this map. The application of this algorithm is shown in an experiment, where the robot attempts to explore and map the whole unknown warehouse environment, which can be useful mainly for rescue operations. Also as shown in the experiments, the motion planning based on the generalised Voronoi diagram handles the cyclic environments correctly.

The last solved task is the group motion planning inspired by biology, especially by social behaviour of insects. It is rather a theoretical proposal how to plan a motion of a group of robots between two given places in an unknown environment based on the generalised Voronoi diagram. This proposal is inspired by the Reynold flocking algorithm for motion planning of a group of entities in which the whole group is planned as one entity, whereas the entities inside this group are driven by three simple rules. The useful information encoded in the generalised Voronoi diagram is a distance to surrounding obstacles in each vertex. This information can be readily exploited especially for the group motion planning in narrow corridors, where the group has to change its shape to prevent a congestion. Lastly in this subject, the formation motion planning is also discussed.

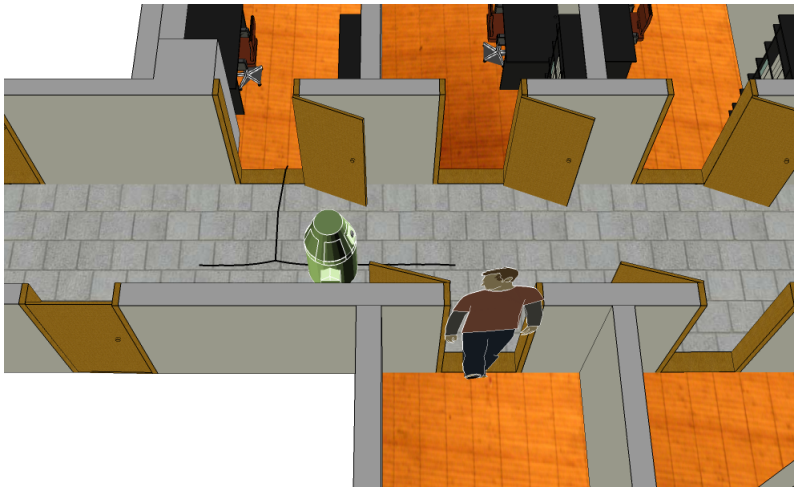

Figure 4.1: Robot motion planning task in an office-like environment with dynamic obstacles. The robot attempts to move into the office while avoiding a person going in the opposite direction.

The group motion planning is an exciting field and can be further expanded for other tasks like the map generation and exploration, include kinodynamic constraints, different sizes and shapes of robots, where even their reconfigurable abilities can be considered.

A simulator for all experiments has been implemented, which includes challenging implementations of different techniques from the computational geometry together with their complex data structures. This work can also serve as a detailed guide to the implementation of the Fortune algorithm including the difficulty with treating degenerate cases of this algorithm and maintenance of its data structures.

The motion planning for dynamic obstacles based on the generalised Voronoi diagram can be developed. A simple manual simulation from the simulator of this type of planning can be found in the thesis itself (see Figure 4.1 for an example). In this simulation, the robot attempts to move into the office from the corridor while avoiding a moving person. This is a challenging task in robotics since the robot must incorporate not only the path computing technique itself but also predictability and reasoning ability.

The approximation algorithms can be also adapted to 3D spaces for motion planning of flying or diving robots. These algorithms would be based on the 3D Voronoi diagram whose geometrically precise representation is even more challenging, therefore using an approximation algorithm would be more than appropriate for practical applications.

There are still a lot of ways of optimising of all the proposed algorithms in future, especially when they are being deployed into a real environment.

REFERENCES

[1] Blanco, D., Boada, B. L., and Moreno, L. Localization by Voronoi Diagrams Correlation. In *ICRA* (2001), IEEE, pp. 4232–4237.

[2] Brauln, T. *Embedded Robotics, Mobile Robot Design and Applications with Embedded Systems.* Springer-Verlag, New York, 2006.

[3] Brock, O., Moll, R., N., and Yang, Y. Efficient and Robust Computation of an Approximated Medial Axis. In *Proceedings of the ACM Symposium on Solid Modeling and Applications* (Italy, 2004).

[4] Choset, H. Incremental Construction of the Generalized Voronoi Diagram, the Generalized Voronoi Graph, and the Hiearchical Generalized Voronoi Graph. In *First CGC Workshop on Computational Geometry* (1997).

[5] Choset, H., and Burdick, J. Sensor Based Motion Planning, Part I: The Generalized Voronoi Graph. In *Proceedings of IEEE/ICRA International Conference on Robotics and Automation* (Nagoya (Japan), 1995).

[6] Choset, H., and Burdick, J. Sensor Based Motion Planning, Part II: Incremental Construction of the Generalized Voronoi Graph. In *Proceedings of IEEE/ICRA International Conference on Robotics and Automation* (Nagoya (Japan), 1995).

[7] Choset, H., and Burdick, J. Sensor Based Motion Planning: The Hiearchical Generalized Voronoi Graph. In *Workshop on Algorithmic Foundations of Robotics* (1996).

[8] Choset, H., and Konukseven, I. Mobile Robot Navigation: Implementing the GVG in the Presence of Sharp Corners. In *Proceedings of IROS* (Grenoble (France), 1997).

[9] Choset, H., Konukseven, I., and Burdick, J. Mobile Robot Navigation: Issues in Implementing the Generalized Voronoi Graph in the Plane. In *Proceedings of IEEE/MFI* (1996).

[10] Choset, H., Konukseven, I., and Rizzi, A. Sensor Based Motion Planning: A Control Law for Generating the Generalized Voronoi Graph. In *Proceedings of IEEE/ICAR* (Montery (CA), 1997).

[11] Choset, H., and Nagatani, K. Toward Robust Sensor Based Exploration by Constructing Reduced Generalized Voronoi Graph. In *Proceedings of the International Conference on Intelligent Robots and Systems* (Kyongju (Korea), 1999).

[12] Culver, T., Hoff III, K., E., Keyser, J., Lin, M., C., and Manocha, D. Interactive Motion Planning Using Hardware-Accelerated Computation of Generalized Voronoi Diagrams. In *Proceedings of IEEE International Conference on Robotics and Automation* (2000), pp. 2931–2937.

[13] De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Utrecht, Netherlands, 2000.

[14] Decastro, L. N., Zuben, F. J. V., Pub, I. G., and Castro, L. N. D. *Recent Developments In Biologically Inspired Computing*. IGI Publishing, Hershey, PA, USA, 2004.

[15] Fortune, S. A Sweepline Algorithm for Voronoi Diagrams. In *Second Annual Symposium on Computational Geometry* (New York, 1986), pp. 313–322.

[16] Fortune, S. Voronoi Diagrams and Delaunay Triangulations. In *Computing in Euclidean Geometry* (Singapore, 1995), vol. 4 of *Lecture Notes Series on Computing*, pp. 225–265.

[17] Fulcher, J. *Advances in Applied Artificial Intelligence (Computational Intelligence and Its Applications) (Computational Intelligence and Its Applications)*. IGI Publishing, Hershey, PA, USA, 2006.

[18] Hoff III, K., E., Keyser, J., Lin, M., C., and Manocha, D. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. In *Proceedings of SIGGRAPH* (1999), pp. 277–286.

[19] Holleman, C., and Kavraki, L. Interactive Motion Planning Using Hardware-Accelerated Computation of Generalized Voronoi Diagrams. In *Proceedings of IEEE International Conference on Robotics and Automation* (2000), pp. 1408–1413.

[20] Kennedy, J., and Eberhart, R. C. *Swarm intelligence.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[21] Konar, A. *Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain.* CRC Press, Inc., Boca Raton, FL, USA, 2000.

[22] Korsh, J. F., and Garrett, L. J. *Data structures, algorithms, and program style using C.* PWS Publishing Co., Boston, MA, USA, 1988.

[23] Latombe, J., C. *Robot Motion Planning.* Kluwer Academic, 1991.

[24] Lavalle, S. *Planning Algorithms.* Cambridge University Press, Cambridge, 2006.

[25] Masehian, E., and Amin-Naseri, M. R. A Voronoi Diagram-Visibility Graph-Potential Field Compound Algorithm for Robot Path Planning. *J. Robot. Syst. 21*, 6 (2004), 275–300.

[26] Muller, H., A., and Wong, K. An Efficient Implementation of Fortune's Plane-Sweep Algorithm for Voronoi Diagrams. In *Technical Report DCS-182-IR.* Department of Computer Science, University of Victoria, 1991.

[27] Nieuwenhuisen, D., Kamphuis, A., Mooijekind, M., and Overmars, M. H. Automatic Construction of High Quality Roadmaps for Path Planning. Tech. Rep. UU-CS-2004-068, Department of Information and Computing Sciences, Utrecht University, 2004.

[28] Okabe, A., Boots, B., Sugihara, and K. Chiu S., N. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, 2000.

[29] Passino, K. M. *Biomimicry for Optimization, Control, and Automation.* Springer, 2005.

[30] Reynolds, C. W. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics 21*, 4 (1987), 25–34.

[31] Roque, W., L., and Doering, D. Constructing Approximate Voronoi Diagrams from Digital Images of Generalized Polygons and Circular Objects. In *Proceedings of the 26th annual Conference on Computer Graphics and Interactive techniques* (Plzen, 2003), pp. 277–286.

[32] Šeda, M. A Comparison of Roadmap and Cell Decomposition Methods in Robot Motion Planning. *WSEAS Transactions on Systems and Control 2*, 2 (2007), 101–108.

[33] Švec, P. Motion Planning of a Swarm of Robots Using Flocking Algorithm and Voronoi Diagram. In *Proceedings of the 39th Spring International Conference Modelling and Simulation of Systems MOSIS '05* (MARQ Ostrava, Hradec nad Moravicí, 2005), pp. 214–220.

[34] Švec, P. Using Flocking Algorithms and Voronoi Diagram for Motion Planning of a Swarm of Robots (Plánování pohybu skupiny robotů pomocí flocking algoritmu a Voroného diagramů). In *Proceedings of the XXXth ASR 2005 Seminar Instruments and Control* (VŠB-TU, Ostrava, 2005), pp. 63 + 449–455.

[35] Švec, P. A Construction of the 2D Generalized Voronoi Diagram, Part I: An Approximation Algorithm. In *Proceedings of the 12th International Conference on Soft Computing MENDEL 2006* (Brno, Czech Republic, 2006), pp. 124–134.

[36] Švec, P. A Construction of the 2D Generalized Voronoi Diagram, Part II: Some Issues of Implementation of Fortunes Plane Sweep Algorithm. In *Proceedings of the 12th International Conference on Soft Computing MENDEL 2006* (Brno, Czech Republic, 2006), pp. 135–144.

[37] Švec, P. *Using Methods of Computational Geometry in Robotics*. PhD thesis, Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Brno, 2007.

[38] Waringo, M., and Henrich, D. Efficient Smoothing of Piecewise Linear Paths with Minimal Deviation. In *IROS 2006, IEEE/RSJ International Conference on Intelligent Robots and Systems* (Beijing, China, 2006), pp. 3867–3872.

Petr Švec

Date of Birth: 20 March 1980
Nationality: Czech
E-mail: petr.svec.cz@gmail.com
Phone: +420 607 613 350

Lidická 79
Hrušovany nad Jevišovkou
671 67
Czech Republic

Education and Qualifications

2003 – 2007 **Ph.D. degree in Robotics,** Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology
Ph.D. Thesis: Using Methods of Computational Geometry in Robotics

2005 – 2006 **Ph.D. student, research fellowship,** University of Bristol, Faculty of Engineering, Department of Computer Science
Project: Using Methods of Computational Geometry in Robotics

2004 – 2006 **Master‘s degree in Computer Science,** Applied Informatics, Faculty of Informatics, Masaryk University Brno
I was within 14 % of the best postgraduates of the study programme in the last two years according to the average grade.

1999 – 2004 **Bachelor’s degree in Computer Science,** Applied Informatics, Faculty of Informatics, Masaryk University Brno
I was within 11 % of the best graduates of the study programme according to the average grade.

1998 – 2003 **Master‘s degree in Computer Science,** Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology

Teaching Experiences

2003 – 2005 **Engineering Computer Science and Automation, Faculty of Mechanical Engineering, Brno University of Technology**
Seminar on Graph Theory – teaching of Master’s students
Seminar on Introduction to Design of Algorithms – teaching of First class students

Employment Details

2006 – Present **Software Developer, Wireless Data Services Ltd, Poole, England, since September on sabbatical**

2002 – 2003 **Software Engineer, Computer Analyst and Programmer**, **Calyx Ltd, Brno**

Summer 2003 **Computer Analyst and Programmer, DELINFO Ltd, Brno**

Other Skills and Qualifications

Languages: English (fluent), Czech (native speaker)
English language certificates: University of Bristol English for Academic Purposes certificate, University of Bristol Academic Writing and Oral Skills for Research Purposes certificate, University of Bristol English for Business and Professional Purposes certificate.

Abstract

This thesis deals with the motion planning of an autonomous robot in an unknown or partially known indoor or outdoor environment. This mainly includes tasks of the global or real-time robot motion planning between two given positions. The generated path is based on the generalised Voronoi diagram, which preserves the largest possible distance from surrounding obstacles and topologically captures the continuity of the whole space. Furthermore, the map generation and exploration and group motion planning tasks are solved. Some of these tasks also take into consideration kinodynamics properties of the robot, its localisation, and uncertainty of the environment.

Two novel approximation geometric algorithms for computing the generalised Voronoi diagram based on the Fortune plane sweep technique are proposed whose efficiency is proved theoretically and by accomplished experiments. These algorithms present a trade-off between the efficiency of computation, implementation difficulty, and robustness. Their asymptotic time complexity is $O(n \log n)$ for n input segment generators provided that the approximation precision is upper bounded.

Further, a new geometric algorithm for solving the map generation and exploration task is designed, which exploits the main properties of the generalised Voronoi diagram. In this task the robot must be able to autonomously explore the whole environment and generate its map.

Finally, a proposal of a technique for the motion planning of a group of robots, which uses the maximal clearance information of each vertex of the generalised Voronoi diagram, is discussed. This is inspired by the Reynold flocking algorithm for transferring a group of entities by considering them as one unit in which these entities are driven by three simple rules.