Practical Strategies for Hypotheses Elimination on the Self-Localization Problem

Martin Buck Dirk Schäfer

Hartmut Noltemeier

Report No. 236 August 1999

Abstract

We take a look at the second part of the robot-selflocalization-problem. The hypotheses generated in a solution of the first part of the problem will be efficient reduced with the movement of the robot. A practical approach is described, using realistic paths and imprecise sensors. It operates on voronoi edges and voronoi vertices and can handle polygons with indiscribed obstacles. A new decision strategy will be discussed different from strategy MDL. Estimations will be given for time and space complexities and for the competitve ratio.
Keywords: autonomous robots, self-localization problem, navigation

Preprint-Series
Department of Mathematics and Computer Science
University of Würzburg

Practical Strategies for Hypotheses Elimination on the Self-Localization Problem

Martin Buck Hartmut Noltemeier Dirk Schäfer

University of Würzburg, Chair of Computer Science I, Am Hubland, 97074 Würzburg
E-mail: **{buck,noltemei,disc}@informatik.uni-wuerzburg.de**

1 Introduction

In future, cleaning robots may be available to purge factories, supermarkets and our livingroom. But this simple task requires teamwork of different sensors and different software. The sensors must recognize static and dynamic obstacles not recorded in the given map of the robot. If not, the robot will hurt human customers or damages inventory like standing palettes of food. An important problem in robotics is the self-localization problem. During each night the robot has to wake up and begin to clean its environment. This is easy if the robot starts its task on the same position every day. But if not, the task becomes very difficult. The robot has to localize itself in a first step and has to eliminate similar positions called hypotheses in the second step before it can move to its starting position.

The first part of the problem, called localization problem, was first introduced by Guibas et al. [GMR97]. They solved the problem using a discretization of the map in regions with similar visibility, so-called visibility-cell-decomposition. This problem got a few solutions, for example using a statistical approach [BCFT98] or a practical visibility approach using polygon metrics [KW99]. In this paper we use our feature-based localization to generate the hypotheses set but other approachs can be used as well. [KS99]

All solutions generate a set of hypotheses, but they do not reduce this set to the true position. In past, Dudek et al. [DRW95] gave a theoretical solution using overlay-arrangements. But this solution was very expensive concerning time complexity of $\mathcal{O}(k^2n^4)$. Therein k ist the number of hypotheses and n is the number of vertices of the map polygon. The approach was improved 1996 by Schuierer who uses windows on which the demanded sensor points lie. This approach has a complexity of $\mathcal{O}(kn^2)$. Today, neither a theoretical nor a practical implemented algorithm exists solving this problem.

In the next section we recall the basics used in this report. In section 3 we describe the solution of Schuierer and give some adaptions. Section 4 models a practical approach and shows the problems appearing with it. Section 5 describes more realistic strategies eliminating hypothetical positions. In the last section we give a summary of the paper.

2 Definitions and Basics

Before defining the problem, we have to describe the sensors and the robot itself used in this scenario.

2.1 Maps and visibility

Most important for the problem and the robot is the environmental map. It gives us the model features in the feature-based localization which are mapped onto scan features. But for motion-planning it is important, too. It gives us the valid paths on which the robot moves.

(2.1) **Definition** (polygonal map)
A *polygonal map* K is described as a simple polygon R and a set $\mathcal{H} = \{H_1, \ldots, H_k\}$ of obstacles in its interior. The border of H_i and the border of R are pairwise disjoint. The *interior of the map* is the interior of R without the completed obstacle polygons $H_i \in \mathcal{H}$.

An important concept is denoted by the term window in a map. Windows are later used to determine the sensing points of the robot at which hypotheses can be eliminated. A robot behind the reflex vertex v_i can see this to a window associated vertex at the moment the robot moves over the window (see figure 1).

(2.2) **Definition** (window)
The segment between a reflex vertex $r_{i,j}$ and the next intersection point with the border created from a line with origin v_i over the mutually visible reflex vetrex $r_{i,j}$ is called *window to vertex* v_i.

2.2 The robot's sensors and modules

The robot uses a lot of sensors for perception, object-recognition and movement. In this chapter, we define the motion manager and the scan performed by the laser range finder.

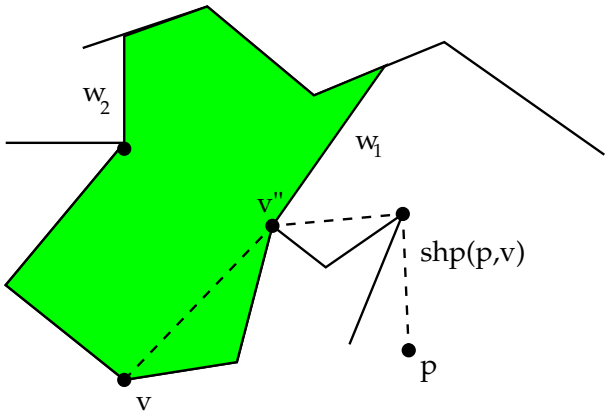

Figure 1: window

A *laser range finder* performs a set of scans at a given position. It is onboard of the robot and typically a SICK-scanner.

(2.3) **Definition** (scan)
A *scan* $S = (R, s)$ is described as a range $R \in \mathbb{R} \cup \{+\infty\}$ and a function $s : [0, 2\pi[\rightarrow [0, R] \cup \{+\infty\}$ with finite domain. If $R = +\infty$, the robot has unlimited range.

(2.4) **Definition** (motion trajectory)
A *motion trajectory* W is a polygonal chain, described as alternating list of angles α_i and lines s_i. $|W|$ denotes the number of lines in W, called *steps of the robot*.

A *motion manager* moves the robot along a motion trajectory W. If the robot reaches the target, the motion manger delivers the endpoint of the trajectory. If not, the motion manager delivers the point it supposes to stand on. During motion, the motion manager tries to compensate the errors of imperfect movement.

2.3 The robot and his job

Giving the main attributes of the robot, we have to define the robot itself and the problems it should solve.

(2.5) **Definition** (autonomous mobile robot)
A robot is defined as a circle with a diameter r_r and a lot of sensors permitting interaction with its environment. We assume that the robot has a laser-range-finder and a compass. If the robot is able to decide its actions only with its sensors, it is called autonomous. If the robot can move freely inside it's environment, it is called mobile.

(2.6) **Definition** (self-localization problem)
An autonomous mobile robot stands at an unknown position in an indoor environment. The robot has a polygonal map of the scene, a compass with low angle adjustment and a laser range finder with unlimited range. Demanded is the position of the robot in the environment, which can be found by scanning the environment and moving the robot.

(2.7) **Definition** (localization problem)
The first subproblem is the localization problem. Using only its compass and its laser range finder the robot has to generate a set of hypothetical positions of its locality at which it may be located.

This problem was solved in many variations (i. e. see [KNSW97][GMR97][BCFT98]). We use our feature-based approach to solve this problem which generates the desired set of hypothetical positions [KS99].

(2.8) **Definition** (navigation problem)
The navigation problem has the same prerequisites as the localization problem. In addition it has a set of hypothetical positions $\mathcal{H}_f$. The problem is to eliminate all wrong positions and to find the true inital robot position at the end.

2.4 Geometric tree and the overlay tree

The above problem will be discussed in section three. Now we define geometric graphs and trees used for modelling the robots paths.

(2.9) **Definition** (geometric graph)
A *geometric graph* is a graph $G = (\mathcal{V}, \mathcal{E})$ with a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$. Thereby every $v \in \mathcal{V}$ is a point of the euclidean plane $\mathbb{R}^2$ and every $e \in \mathcal{E}$ a three tuple $e = (v_1, v_2, C)$ with $v_1, v_2 \in \mathcal{V}$. $C \subset \mathbb{R}^2$ is a curve with endpoints v_1, v_2 and the length $l(C)$.

A *geometric tree* is a geometric graph fulfilling the properties of a tree.

(2.10) **Definition** (subpath)
A *subpath* $W \subset G$ of a geometric graph is a sequence of w edges $(e_1, \ldots e_w)$ and for every $e_i = (v_{i,1}, v_{i,2})$ we have $v_{i,2} = v_{i+1,1}$.

(2.11) **Definition** (length of a path)
For the length $l(W)$ of a *subpath* $W = (e_1, \ldots e_w)$ the equation holds

$$l(W) = \sum_{i=1}^{w} e_i$$

(2.12) **Definition** (overlay tree)
An *overlay tree* is a geometric tree, generated by uniting several trees. It is defined as 5-tuple $B^O = (O, \mathcal{B}, \mathcal{S}, \kappa, \lambda)$ with

- $O = (\mathcal{V}_O, \mathcal{E}_O)$ is a geometric tree
- $\mathcal{B}$ is a set of geometric trees.
- $\mathcal{S}$ is a set of vertices with the property: there is a bijective mapping $\phi : \mathcal{S} \to \mathcal{B}$ and $\phi(s) = \mathcal{B} \Rightarrow s \in \mathcal{B}$. $\mathcal{S}$ is called set of start vertices.
- κ is a function $\kappa : \mathcal{V}_O \to 2^{\mathcal{B}}$, giving for every vertex $v \in O$ the originating tree.
- λ is a function $\lambda : \mathcal{E}_O \to 2^{\mathcal{B}}$, giving for every edge $e \in O$ the originating tree.

All trees $B \in \mathcal{B}$ will be placed successively onto the origin with respect to their start vertex s. All overlapping edges are unified. If two branches disperse, the adjacent trees will be inserted into the overlaytree. Later branches with overlapping property are not unified.

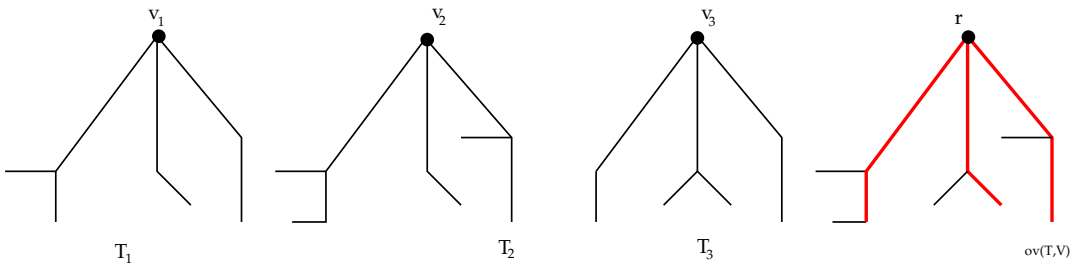

Figure 2: Example of an overlay-tree

Some edges in the overlytree have extra properties:

(2.13) **Definition** (imperfect vertex)
A vertex v with $\kappa(v) \neq \mathcal{B}$ is called an *imperfect vertex*. The set $\mathcal{V}^u = \{v \in \mathcal{V}_O : \kappa(v) \neq \mathcal{B}\}$ is the set of imperfect vertices.

(2.14) **Definition** (nearest imperfect vertex)
The *nearest imperfect vertex* with respect to the origin is the vertex v_{nu} fulfilling the inequation

$$\forall v \in \mathcal{V}^u \, : \, l(W_{v_{nu}}) \leq l(W_v)$$

2.5 Valid robot paths

So far, geometric graphs and overlay trees are defined and are used as essential data structures. They will be used as travel paths for the robot by using shortest-path-trees. Another possibility is to use the voronoi diagram or a combination of both approaches.

(2.15) **Definition** (shortest-path-tree)
A tree $B(p) = (\mathcal{V}_B, \mathcal{E}_B)$ is a *shortest-path-tree* with respect to a geometric graph, $G = (\mathcal{V}_G, \mathcal{E}_G)$, if

- $B(p)$ is a geometric tree with root $p \in \mathcal{V}_B$.
- $\mathcal{V}_B = \mathcal{V}_G$,
- $\forall v \in \mathcal{V}_B$: the path $W(p, v) \in B$ from p to v exists and $W(p, v)$ is a shortest-path from p to v in G, so $l(W(p, v))$ is minimal with respect to all paths in G.

(2.16) **Definition** (voronoi-skeleton)
A *voronoi-skeleton* of the map K is a voronoi-diagram of K restricted to edges and nodes lying in the interior of K and not adjacent to a reflex vertex.

(2.17) **Definition** (property of the voronoi diagram)
For every element of the voronoi diagram all of the appropriate region can be seen.

(2.18) **Definition** (voronoi path)
A *voronoi path* is a subpath of the voronoi-skeleton. Voronoi vertices of the voronoi path represents sensor nodes for localizing the robot.

2.6 Competitive ratio

We need a measure for the quality of an approximation algorithm giving a solution for $\mathcal{NP}$-complete problems. One measure for an online-algorithm is the competitve ratio introduced by Sleator and Tarjan 1985 [ST85]. Before we introduce this measure for the problem, we have to define a decision strategy and a measure for the path-length of the robot-path.

(2.19) **Definition** (decision strategy)
A *decision strategy* determines in which order the sensor nodes will be visited and therby in which order hypothesis will be eliminated.

(2.20) **Definition** (verification path)
A robot stands at point p inside the map K. It uses a localization algorithm L to generate a set of hypothetical positions $\mathcal{H}_f$. To determine the position of the robot, it needs to eliminate the hypotheses set. For this it uses a decision strategy N and travels along a *verification path* $W_{L,N}(p, P)$ to the determined sensor nodes.
The optimal verification path is denoted by $W^*_{L,N}(p, P)$.

(2.21) **Definition** (competitive ratio)
The *competitive ratio* $C_{L,N}$ depending on localization algorithm L and decision strategy N is the ratio

$$C_{L,N} = \max_{P \in \mathcal{P}, p \in P} \frac{l(W_{L,N}(p, P))}{l(W^*_{L,N}(p, P))}$$

Thereby $\mathcal{P}$ is the set of all map polygons. An approximate solution of the problem has a *competitive ration* of $C_{L,N}$ times the optimal solution.

Descriptive spoken the competitive ratio belongs to the length of the path caused by the localization strategy L and the decision strategy N in a worst-case map on worst-case positions and the optimum path for this worst-case scenario.

3 The approach of Schuierer and adaptations

The second part of the self-localization problem was first introduced and solved by Dudek et. al [DRW95]. They showed that the problem is $\mathcal{NP}$-complete by reducing it on the minimum-decision-tree problem. They gave an approximative solution by using overlay-arrangements of visibility decompositions. Their algorithm has a time-complexity of $\mathcal{O}(k^2n^4)$ whereas k denotes the number of hypotheses and n denotes the number of vertices of the map.

In 1996 Schuierer gives a better solution with a time-complexity of $\mathcal{O}(kn^2)$. He uses windows instead of a visibility decomposition of the map. Then he computes

shortest-path-trees for all hypothetical positions on every position to all vertices of the map. The sensor points for eliminating wrong hypotheses and also the nodes of the shortest-path-tree are defined as the intersection points of the shortest-path to a desired vertex v_i of the polygon with the dedicated window of v_i. This shortest-path-trees are stored in an overlay tree. In this tree the decision strategy searches for imperfect vertices which symbolize the difference in all the trees and hence a usefull sensing position. Like Dudek, Schuierer uses the decision strategy *minimum-distance-localization MDL* to reduce hypothetical positions which select always the nearest imperfect vertex as the next sensing node. This greedy-strategy has a competitive ratio of $2(k-1)$ by returning to the origin after each reduction step. Descriptive spoken, if in every step only one hypothesis would be eliminated, up to $(k-1)$ steps must be made. The factor 2 comes from returning to the origin each time.

Algorithm 1 Schuierers algorithm

1 A localization algorithm gives a set $\mathcal{H}_f$ of hypothetical robot positions.
2 Compute for each vertex v in P and every $h \in \mathcal{H}_f$ the sensing point v_h^*. v_h^* is the nearest point to h lying on the nearest window from whom v is visible.
3 Compute for every hypothesis $h \in \mathcal{H}_f$ a shortest-path-tree B_h from the root h to all sensing points v_h^*. So $\mathcal{B} = \{B_h \mid h \in \mathcal{H}_f\}$.
4 Create the overlay tree $B^O = (O, \mathcal{B}, \mathcal{H}_f, \kappa, \lambda)$.
5 **while** $\mid \mathcal{H}_f \mid > 1$ **do**
6 b^* is the nearest imperfect vertex in B^O
7 Calculate a shortest path W_{b^*} from the origin to b^*.
8 Follow the path W_{b^*}, take a sensing step and return to the origin.
9 Reduce hypotheses $\bar{h}$ not consistent with the sensor information and update the overlay tree.
10 **end while**

3.1 Using polygonal obstacles

Schuierer mentioned polygonal obstacles in his approach but neither he gave details for implementation nor he provide complexitys and competitive ratio of this advanced problem.

(3.22) **Corollary** (Approach with polygonal obstacles)
Using Schuierers algorithm and allowing simple polygonal obstacles inside the map K, the time-complexity increases to $\mathcal{O}(kn^2 \log(n))$.

Proof: For the algorithm only the number of windows are important. Including obstacles, there are still $\mathcal{O}(nr)$ windows. But now n denotes the number of vertices of the map polygon and the obstacle polygons.

The linear triangulation algorithm described by Chazalle in [Chaz91] cannot be used in this case, because it only operates on simple polygons without obstacles. So the triangulation time increases up to $\mathcal{O}(n(r+1))$ by using data structures as discribed in [Held99].

The computation of shortest-path-trees can be done by using Dijkstras algorithm with respect to non-negative edge weighting and cycles. This algorithm has a complexity of $\mathcal{O}(n^2 \log(n))$.

Rayshooting can performed with the same complexity as before by using data structures described in [CPT93] with complexity of $\mathcal{O}(n \log^3(n))$.

Altogether the algorithm has a complexity of $\mathcal{O}(kn^2 \log(n)$.

□

(3.23) **Corollary** (Space complexity with obstacles)
Using Schuierers algorithm and allowing simple polygons as obstacles inside the map K, the space complexity is $\mathcal{O}(n^2 \log n)$.

Proof: Storing all windows costs $\mathcal{O}(rn)$ storage. Thereby r denotes the number of reflex vertices. Storing the triangulation of the polygon costs $\mathcal{O}(n^2)$ storage [Held99]. After this, we need k shortest-path-trees each with a storage of $\mathcal{O}(n \log n)$ so it sums up to $\mathcal{O}(kn \log n)$. After all we need storage for the rayshooting with complexity of $\mathcal{O}(n \log n)$. Altogeter the memory usage doesn't succed $\mathcal{O}(n^2 \log n)$.

□

(3.24) **Corollary** (Competitive ratio with obstacles)
Using Schuierers algorithm and allowing simple polygons as obstacles inside the map K, the competitive ratio is $2(k-1)$, thereby $k \leq \sqrt{n}$.

Proof: Dijkstras algorithm computes a shortest path even in the case of obstacles. If there is a shorter path $W^*_{L,N}(p^*, P)$ from the origin to a sensing point p^*, then p^* is a imperfect vertex of the overlay tree. But this is a contradiction to strategy MDL who select point p as the nearest point to the origin.

We only use shortest-path-trees inside the overlay tree. If we have k hypothetical positions then we need at most $(k-1)$ eliminating steps to find the correct robot position. If the robot moves back to the origin in each step, the competitive ratio becomes $2(k-1)$.

□

4 Modelling a practical approach

Schuierers algorithm improves the time complextity by comparison with Dudeks approach, but his approach is theoretical. In practice, we need a few adaptations that are presented in this section. We need a realistic robot, realistic robot trajectories and practical decision strategies for a less idealistic approach.

4.1 Spacial spread of the robot

For a practical approach it is necessary to have a robot with a spacial spread and can not assume a point-like robot. So the robot is circular with a diameter r and uses a safety margin ε to all obstacles and walls.

This leads to constraints for the given map. The robot must have the ability to see every point of the map. Further the robot is not imprisoned in a part of the map. We can formulate it in terms of visibility: the robot must have the ability to move to all points so that every point of the map is mutually visible with it.

4.2 Using modified shortest-paths

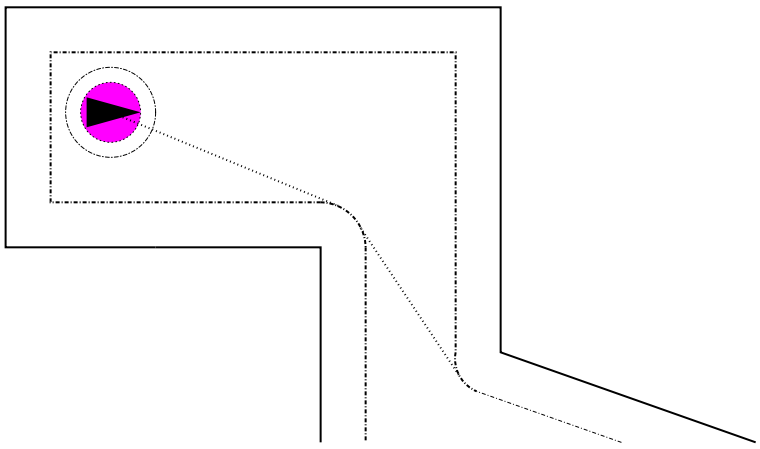

Figure 3: Real shortest-path with safety margin and robot diameter

Fulfilling the practical approach and permit the robot to move only in secure regions of the map, we can use Schuierers algorithm. In this case an adaptation of the shortest-path computation is necessary. Edges of the shortest-paths may only use secure regions so sensor points ly on windows a safety distance away from any vertex of a map polygon. Also the verification-path-length of the modified shortest-path-tree is longer than the original verification-path-length.

Problems with this sort of path arise for the recognition and tracking of the verification path by the robot. The laser range finder is only a low-cost sensor system, exact navigation with this system is impossible. So we need another approach for path planning.

4.3 Using voronoi-paths

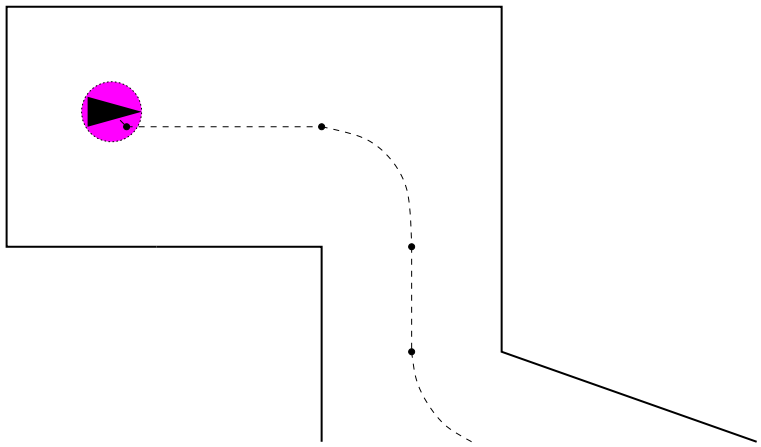

Figure 4: Used voronoi path

Voronoi-paths are an alternative approach. Voronoi paths are the locally securest paths for a robot. They have the poperty of maximum clearance to all obstacles in the environment. Another positive attribute is their very fast computability. In addition the sensor system may be able to navigate on voronoi-paths.

Moving to the nearest voronoi-node
After localization and hypotheses generation the robot must determine the nearest voronoi-node and move onto it. This task is not involved in the overlay tree and has to be done from a realistic motion manager. In this research report we assume that the robot can reach the next voronoi vertex without any errors. In addition we assume that the robot does not stand exactly between two voronoi-nodes and cannot determine which node is the nearest.

The maximum distance the robot needs to reach the nearest voronoi vertex is $\frac{1}{2}\sqrt{2}m$. Thereby m denotes the length of the largest voronoi edge in the map (see figure 5).

Moving along voronoi-paths
For the movement along voronoi paths there exists no estimation of the ratio of covered path-length, because in the worst-case the ratio is unrestricted. We can only give an estimation for parabolas with respect to the shortest-path. There the

Algorithm 2 Advanced voronoi algorithm

```
1  Create the voronoi diagram
2  For every h_i ∈ H_f compute the next visible voronoi vertex v_i
3  For all v_i compute the shortest-path-tree B_i with respect to the voronoi
   diagram
4  Build the overlay tree B^O by inserting all B_i with origin v_i
5  Move the robot from its initial position h_0 to the voronoi node v_0.
6  while |H_f > 1| do
7    Compute the nearest imperfect vertex v_i^nu
8    Compute the shortest-path W_i to v_i^nu in B^O
9    while W_i contains unvisited segments do
10     Follow the next segment s and take a scan at the target node
11     if at least one hypothesis can be eliminated then
12       move back to the origin and finish while-loop
13     end if
14   end while
15   Discard all wrong hypotheses of H_f and update B^O
16 end while
```

robot travels only $\sqrt{2}$ times longer than the shortest path [Buck99] connecting the adjacent voronoi vertex.

The unrestricted ratio cannot be estimated efficiently and therefore we can only give an idea for a solution. We must extend the voronoi diagram by inserting shortcuts, if a path between v_1 and v_3 is much shorter than a path beween v_1 over v_2 to v_3. These shortcuts must have the property to be recognizable and trackable by the robot using its laser range finder. Too many shortcuts inserted will raise the time complexity of the algorithm and are therefore not useful. The problem is not solved and has to be examined in future work.

4.4 Competitve ratio and complexity of the practical approach

Despite the problem in the previous section we will examine the competitve ratio and the complexity of the advanced voronoi-algorithm. Shortcuts are not covered by our complexity estimation.

(4.25) **Theorem** (Time complexity of the voronoi algorithm)
The voronoi algorithm has a time complexity $\mathcal{O}(kn\log(n))$ with polygonal obstacles inside the map K.

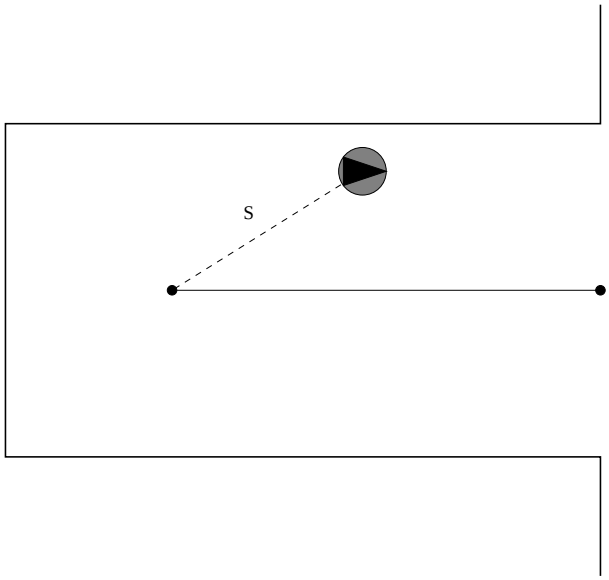

Figure 5: Moving to the nearest voronoi-node

Proof: Calculation of the voronoi-diagram is performed in $\mathcal{O}(n \log(n))$ with obstacles inside the map. Calculation of the nearest voronoi-node can be done for every hypothesis in $\mathcal{O}(kn \log(n))$ with eligible data structures (see ([CPT93]).

Building up the shortest-path-trees can be done in $\mathcal{O}(kn \log(n))$ for all hypothesis. Constructing the overlaytree uses $\mathcal{O}(n \log(\delta))$, where $\delta \leq kn$ is the maximum degree of a node in B^O [Schu96].

Computing the path to the nearest voronoi vertex can be done in $\mathcal{O}(1)$ under the assumption of an ideal motion manager. In realistic scenes there will be a large additional expense, depending on the distance, the robot has to travel.

The next imperfect vertex in the overlay tree can be computed in $\mathcal{O}(n)$, because there exist $\mathcal{O}(n)$ voronoi vertices. The calculation of the shortest path to an imperfect vertex has a complexity of $\mathcal{O}(n)$, because of the number of $\mathcal{O}(n)$ voronoi edges in the voronoi diagram.

The movement along voronoi paths may be ignored for now because of the assumtion of an ideal movement. Essential for the complexity is the sensing at voronoi vertices and the search in the set of hypotheses until at least one hypothesis can be eliminated. In the innermost loop an effort of $\mathcal{O}(k \log(k))$ has to be account and for the total loop an effort of $\mathcal{O}(nk \log(k))$. Remark that sensing costs are neglected.

Altogether summes up to a running time of $\mathcal{O}(kn\log(n))$.

□

(4.26) **Theorem** (Competitive ratio of the voronoi algorithm)
The voronoi-algorithm with obstacles has a competitve ratio of $2(k-1)$ *times the optimum verification path restricted to voronoi paths.*

Proof: If we assume that the map of the robot fulfills the described constraints, every point of the polygon can be seen from a voronoi vertex.

In the algorithm there is no change exept the restriction only moving along voronoi paths. These are generated from shortest-path trees, too.

In every step of the algorithm we eliminate at least one hypothesis ore move back to the origin using voronoi paths. The path to the optimal sensing position is restricted to voronoi paths, so there is no extra detour in the algorithm.

We use $(k-1)$ reduction steps in the worst-case and in the optimum case it is done in one step so the competitve ratio is $2(k-1)$ under the assumption of moving back to the origin in every step.

□

5 Practical decision strategies

So far, only the decision strategy MDL was presented and analyzed. This strategy has the proofable best competitive ratio. But there are other possibilities of decision strategies that may have a better performance in practice with the same or worser competitive ratio.

5.1 Decision strategy RPL

Here we will present a decision strategy called *residual path length (RPL)*. This strategy does not use the shortest path to the next imperfect node but considers the expected size of the sets of hypotheses to be eliminated. It assumes that all hypotheses are equally distributed and uses a weighting function by multiplying the length of a path to the next sensing point with the expected value of the size of the hypotheses set. The weighting function is described as

$$\forall i \ : \ \overline{l(W)_i} = l(W_i) \cdot \frac{1}{k} \cdot \overline{|\mathcal{H}_f|}_i \tag{5.27}$$

thereby $\overline{|\mathcal{H}_f|}_i$ is the *expected size of the hypotheses set*. It can be estimated as

$$\overline{|\mathcal{H}_f|}_i \leq \frac{\left|\mathcal{H}_{f_{1,i}}\right|}{|\mathcal{H}_f|}\left|\mathcal{H}_{f_{1,i}}\right| + \frac{|\mathcal{H}_f| - \left|\mathcal{H}_{f_{1,i}}\right|}{|\mathcal{H}_f|}(|\mathcal{H}_f| - \left|\mathcal{H}_{f_{1,i}}\right|) \tag{5.28}$$

It is a safe estimation because the hypotheses set moulders into a number of indefinitly sets and in two sets in worst-case. The term $\frac{1}{k}$ is necessary for standardization.

In every step the strategy searches for the minimum value of the weighting function (5.27) of all members in the set of imperfect nodes. Remark that the generation of the sensing points is the same as in strategy MDL.

5.2 Decision strategy IEE

If the set of generated hypotheses is not equally distributed, than we need another adapted strategy called *inverse expected entropy (IEE)*. This strategy assumes that a localization algorithm produces a set of ranked hypotheses like the feature-based localization. So the hypotheses have different probabilities according to the ranking position and a weighting function.

Therefore we have to replace the equation for the expected value of the hypotheses size by the equation that describes the information content of the hypotheses.

$$\overline{|\mathcal{H}_f|}_i \leq \frac{\sum p(h_j)}{\sum p(h_g)} \cdot \left|\mathcal{H}_{f_{1,i}}\right| + \frac{\sum p(h_k)}{\sum p(h_g)} \cdot (|\mathcal{H}_f| - \left|\mathcal{H}_{f_{1,i}}\right|) \tag{5.29}$$

Thereby $h_j \in \mathcal{H}_{f_{1,i}}, h_k \in (\mathcal{H}_f - \mathcal{H}_{f_{i,1}}), h_g \in \mathcal{H}_f$.

Like RPL, this strategy searches for the minimum of the weighting function (5.27) in the set of imperfect nodes, too. There are only small adaptations needed on the presented voronoi-algorithm or Schuierers algorithm to realize the new strategies so we disclame a new representation.

5.3 Proof of the competitive ratio

In this section we will proof the competitive ratio for a decision strategy who uses the same set of sensor nodes as strategy MDL and also the minimum value of a weighting function in every step.

(5.30) **Theorem** (Competitive ratio of a weighting function approach)
We assume that a decision strategy for hypotheses elimination uses in every step the minimum value of a weighting function $s_i = h_i \cdot l_i$ for an arbitrary i and eliminates at least one hypothesis. Thereby h_i denotes a strategy value and l_i the shortest-path length to a sensing node. Then this strategy has a competitive ratio of $2(k-1)\frac{h_{max}}{h_{min}}$ when returning back to the origin in every step.

Proof: The sensor points and so called imperfect nodes in the overlaytree are computed using shortest-paths. So there is no detour in reaching these points.

Consider that this strategy reaches the right hypothesis at last. Then $(k-2)$ hypotheses will be reached before. Also we get

$$s_1 \leq s_2 \cdots \leq s_{k-2} \leq s_{k-1} \tag{5.31}$$

We can write the inequation as

$$l_1 \cdot h_1 \leq \cdots \leq l_{k-1} \cdot h_{k-1} \tag{5.32}$$

Since l_{k-1} is the optimum shortest path d and h_{opt} the strategy value for this we get the inequations

$$l_i \cdot h_i \leq d \cdot h_{opt} \Rightarrow l_i \leq \frac{d \cdot h_{opt}}{h_i} \tag{5.33}$$

for $1 \leq i \leq k-2$.
For the sum of the verification path we get the estimation

$$\sum_{i=1}^{k-1} l_i \leq (k-1)\frac{d \cdot h_{opt}}{h_{min}} \leq (k-1)\frac{d \cdot h_{max}}{h_{min}} \tag{5.34}$$

So, the competitve ratio gets

$$C_{f,w} = \max_{P \in \mathcal{P}, p \in P} \frac{l(W_{f,w}(p,P))}{l(W^*_{f,w}(p,P))} = (k-1)\frac{h_{max}}{h_{min}} \tag{5.35}$$

So the competitive ratio can be arbitrary worse belonging to a given probability distribution and a weighting function. The competitive ratio is $2(k-1)$ in the case of $h_i = 1$ for all i.

□

For the strategies in the previous section we get worser competitive ratios, too. Here we will give a result for strategy RPL.

(5.36) **Theorem** (Competitive ratio of strategy RPL)
Strategy RPL has a competitive ratio of $4(k-1)$ times the optimum path-length when returning back to the origin in every step.

Proof: Using the above theorem we have to search only for the minimum and the maximum of the weighting function.

From equation (5.27) and equation (5.28) we get a minumum value $\frac{k}{2}$ and a maximum value $(k-1)$. Inserted in the equation we get the minumum value of the weighting function $h_{min} = \frac{k}{2}$ and the maximum value $h_{max} = k$.

So the competitive ratio gets $4(k-1)$ when returning back to the origin in every step.

□

For a practical approach we hope that the presented strategies doing better than MDL because the competitive ratio measures only the worst-case. First studies indicate an advantage of MDL only for a small hypotheses set and a disadvantage for huge hypotheses sets (see [Buck99]).

6 Summary

In this report we gave a practical approach for the second part of the self-localizing problem. We expanded Schuierers algorithm to polygons with obstacles. Next we gave a more realistic approach in using voronoi paths. Finally we presented new decision strategies for eliminating the hypotheses set.

Many problems are still unsolved. First we need an estimation or solution for the unrestricted voronoi length ratio. Also we have to investigate the new decision strategies using simulation studies and practical tests, too. The fundamental part in creating a motion manager was only mentioned and must be solved in future.

7 Acknowledement

The authors would like to thank the DFG in supporting this project (No 88/14-2) and also Boris Kluge for his programming support.

References

[BCFT98] W. Burgard, Armin B. Cremers, Dieter Fox and Sebastian Thrun. Position Estimation for Mobile Robots in Dynamic Environments, 1998.

[Buck99] Martin Buck. Effiziente Strategien zur Hypothesenreduktion bei der Lokalisation und Navigation autonomer Roboter. Master's thesis, University of Würzburg, Juli 1999.

[Chaz91] Bernard Chazelle. Triangulating a Simple Polygon in Linear Time. *Discrete Computational Geometry*, 6(1):485–524, 1991.

[CPT93] Yi-Jen Chiang, Franco P. Preparata and Roberto Tamassia. A Unified Approach to Dynamic Point Location, Ray Shooting, and Shortest Paths in Planar Maps. In *4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993.

[DRW95] Gregory Dudek, Kathleen Romanik and Sue Whitesides. Localizing a Robot with Minimum Travel. In *Proceedings of the 6th Annual ACM–SIAM Symposium on Discrete Algorithms*, pp. 437–446, 1995.

[GMR97] Leonidas J. Guibas, Rajeev Motwani and Prabhakar Raghavan. The Robot Localization Problem. *SIAM Journal on Computing*, 26(4):1120–1138, August 1997.

[Held99] Martin Held. FIST: Fast Industial-Strength Triangulation of Polygons. to appear in: Algorithmica, April 1999.

[KNSW97] Oliver Karch, Hartmut Noltemeier, Mathias Schwark and Thomas Wahl. Relokalisation – Ein theoretischer Ansatz in der Praxis. In *Autonome Mobile Systeme 1997 (AMS'97)*, pp. 119–130, 1997.

[KS99] Boris Kluge and Dirk Schäfer. Featurebasierte Lokalisation eines autonomen mobilen Roboters. Technical Report 234, University of Würzburg, 1999.

[KW99] Oliver Karch and Thomas Wahl. Relocalization – Theory and Practice. In Hartmut Noltemeier and Kokichi Sugihara (Editor), *Special Issue of Discrete Applied Mathematics on Computational Geometry*, Volume 93. Elsevier, 1999.

[Schu96] Sven Schuierer. Efficient Robot Self-localization in simple polygons. In *Intelligent Robots*, pp. 129–146, 1996.

[ST85] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 23:202–208, 1985.

Preprint-Reihe
Institut für Informatik
Universität Würzburg

Verantwortlich: Die Vorstände des Institutes für Informatik.

[179] H. Vollmer. *Relating Polynomial Time to Constant Depth.* August 1997.

[180] S. Wahler, A. Schömig, O. Rose. *Implementierung und Test neuartiger Zufallszahlengeneratoren.* August 1997.

[181] J. Wolff von Gudenberg. *Objektorientierte Programmierung im wissenschaftlichen Rechnen.* September 1997.

[182] T. Kunjan, U. Hinsberger, R. Kolla. *Approximative Representation of boolean Functions by size controllable ROBDD's.* September 1997.

[183] S. Kosub, H. Schmitz, H. Vollmer. *Uniformly Defining Complexity Classes of Functions.* September 1997.

[184] N. Vicari. *Measurement and Modeling of WWW-Sessions.* September 1997.

[185] U. Hinsberger, R. Kolla. *Matching a Boolean Function against a Set of Functions.* November 1997.

[186] U. Hinsberger, R. Kolla. *TEMPLATE: a generic TEchnology Mapping PLATform.* November 1997.

[187] J. Seemann, J. Wolff von Gudenberg. *OMT-Script - eine Programmiersprache für objektorientierten Entwurf.* November 1997.

[188] N. Gerlich, M. Ritter. *Carrying CDMA Traffic over ATM Using AAL-2: A Performance Study.* November 1997.

[189] N. Gerlich. *The Performance of Base Station Interconnection Alternatives in CDMA Networks.* Dezember 1997.

[190] M. Ritter. *Discrete-Time Modeling of the Frame-Based Generic Cell Rate Algorithm.* Januar 1998.

[191] S. Reith, K. W. Wagner. *On High and Low Sets for the Boolean Hierarchy.* Januar 1998.

[192] P. Tran-Gia, N. Jain, K. Leibnitz. *Code Division Multiple Access wireless network planning considering clustered spatial customer traffic.* Februar 1998.

[193] K. Leibnitz, P. Tran-Gia, J. E. Miller. *Analysis of the Dynamics of CDMA Reverse Link Power Control.* Februar 1998.

[194] S. K. Bamberger, S. Schwingeler, S. Ziegler. *KBA-D3: Ein wissensbasiertes Diagnose- und Informationssystem für Druckmaschinen.* Februar 1998.

[195] S. Reith, K. W. Wagner. *On Boolean Lowness and Boolean Highness.* März 1998.

[196] S. Reith, H. Vollmer. *The Complexity of Computing Optimal Assignments of Generalized Propositional Formulae.* März 1998.

[197] C. Glaßer. *Klassifizierung der Nicht-Approximierbarkeit eines Basisstationsproblems mit Interferenzen.* März 1998.

[198] N. Vicari. *Models of WWW-Traffic: a Comparison of Pareto and Logarithmic Histogram Models.* März 1998.

[199] N. Gerlich, M. Menth. *The Performance of AAL-2 Carrying CDMA Voice Traffic.* April 1998.

[200] W. Nöth, R. Kolla. *Spanning Tree Based State Encoding for Low Power Dissipation.* April 1998.

[201] H. Schmitz, K. W. Wagner. *The Boolean Hierarchy over Level $1/2$ of the Straubing-Thérien Hierarchy.* April 1998.

[202] J. Steffan, H.-C. Wirth. *ANDI - Approximation Algorithms for Network Design and Network Improvement.* April 1998.

[203] O. Rose. *WIP Evolution of a Semiconductor Factory After a Bottleneck Workcenter Breakdown.* April 1998.

[204] O. Rose. *Interdeparture Time Correlations of the Discrete-time GI/GI/1 Queue.* April 1998.

[205] M. Galota. *Blattsprachen und endliche Automaten.* Mai 1998.

[206] W. Jodl, M. Heuler, U. Rothaug, K. Leibnitz. *Einsatz von topologieerhaltenden neuronalen Netzen zur Interpretation von Leiterplattendaten.* Juli 1998.

[207] N. Vicari, R. Schedel. *Performance of the GFR-Service with Constant Available Bandwidth.* Juli 1998.

[208] C. Glaßer. *On the approximability of problems for cellular networks.* September 1998.

[209] H. Vollmer. *A Generalized Quantifier Concept in Computational Complexity Theory.* September 1998.

[210] M. Dümmler. *Analysis of the departure process of a batch server queueing system.* September 1998.

[211] N. Gerlich, M. Menth. *A Numerical Framework for Solving Discrete Markov Models Applied to the AAL-2 Protocol.* Oktober 1998.

[212] M.-A. Remiche, K. Leibnitz. *Adaptive Soft-Handoff Thresholds for CDMA Systems with Spatial Traffic.* Oktober 1998.

[213] F. Duckstein, R. Kolla. *Ray Tracing Of Parametric Surfaces Based On Adaptive Simplicial Complexes.* Oktober 1998.

[214] N. Vicari. *Effects of Variations in the Available Bandwidth on the Performance of the GFR Service.* Oktober 1998.

[215] M. Dümmler, A. Schömig. *Using Discrete-time Analysis in the Performance Evaluation of Manufacturing.* November 1998.

[216] C. Glaßer. *A Normalform for Classes of Concatenation Hierarchies.* Dezember 1998.

[217] S. Kosub. *Persistent Computations.* Dezember 1998.

[218] S. Reith, K. W. Wagner. *The Complexity of Problems Defined by Subclasses of Boolean Functions.* Januar 1999.

[219] C. Glaßer, G. Wechsung. *Relativizing Function Classes.* Januar 1999.

[220] H. Schmitz. *Some Forbidden Patterns in Automata for Dot-Depth One Languages.* Januar 1999.

[221] T. Peichl, H. Vollmer. *Finite Automata with Generalized Acceptance Criteria.* Januar 1999.

[222] H. Vollmer. *Uniform Characterizations of Complexity Classes.* Januar 1999.

[223] U. Ehrenberg, K. Leibnitz. *Impact of Clustered Traffic Distributions in CDMA Radio Network Planning.* März 1999.

[224] H. Vollmer. *Was leistet die Komplexitätstheorie für die Praxis?* März 1999.

[225] R. Kolla, A. Vodopivec, J. Wolff von Gudenberg. *The IAX Architecture: Interval Arithmetic Extension.* April 1999.

[226] H. Schmitz. *Generalized Deterministic Languages and their Automata: A Characterization of Restricted Temporal Logic.* April 1999.

[227] M. Dümmler. *Using Simulation and Genetic Algorithms to Improve Cluster Tool Performance.* Mai 1999.

[228] R. Harris, S. Köhler. *Possibilities for QoS in Existing Internet Routing Protocols.* Mai 1999.

[229] F. Heister, R. Müller. *An approach for the identification of nonlinear, dynamic processes with Kalman Filter-trained recurrent neural structures.* Mai 1999.

[230] C. Glaßer. *The Boolean Hierarchy over Dot-Depth* $1/2$*.* June 1999.

[231] M. Dümmler. *Simulation und Leistungsbewertung von Cluster Tools in der Halbleiterfertigung.* Juli 1999.

[232] C. Glaßer, S. Reith, H. Vollmer. *Approximation Algorithms for Cellular Network Optimization.* Juli 1999.

[233] S. Kosub, K. W. Wagner. *The Boolean Hierarchy of Partitions.* Juli 1999.

[234] B. Kluge, D. Schäfer. *Featurebasierte Lokalisation eines autonomen mobilen Roboters.* Juli 1999.

[235] M. Dümmler, N. Vicari. *A Numerical Analysis of the $M/D^b/N$ Queueing System.* Juli 1999.