DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
45

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
45

PATH PLANNING AND LEARNING STRATEGIES FOR MOBILE ROBOTS IN DYNAMIC PARTIALLY UNKNOWN ENVIRONMENTS

KRISTO HEERO

TARTU UNIVERSITY
PRESS

Faculty of Mathematics and Computer Science, University of Tartu, Estonia

Dissertation is accepted for the commencement of the degree of Doctor of Philosophy (PhD) on May 12, 2006, by the Council of the Faculty of Mathematics and Computer Science, University of Tartu.

Opponent:
Prof. Paolo Fiorini
University of Verona
Verona, Italy

Commencement will take place on June 19, 2006.

ISSN 1024–4212
ISBN 9949–11–307–5 (trükis)
ISBN 9949–11–308–3 (PDF)

Tartu Ülikooli Kirjastus
www.tyk.ee
Tellimus nr. 252

CONTENTS

LIST OF ORIGINAL PUBLICATIONS

1. K.Heero, U.Puus, J.Willemson. XML based document management in Estonian legislative system. In *Proceedings of the 5th International Baltic Conference on DB and IS*, pages 321-330, Tallinn, Estonia, June 2002.
2. M.Kruusmaa, J.Willemson, K.Heero. Path Selection for Mobile Robots in Dynamic Environments. In *Proceedings of the 1st European Conference on Mobile Robots (ECMR'03)*, pages 113-118, Radziejowice, Poland, September 2003.
3. K.Heero, J.Willemson, A.Aabloo, M.Kruusmaa. Robots Find a Better Way: A Learning Method for Mobile Robot Navigation in Partially Unknown Environments. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 559-566, Amsterdam, The Netherlands, March 2004.
4. K.Heero, A.Aabloo, M.Kruusmaa. On The Utility Of Exploration On Time-Critical Mobile Robots Missions. In *Proceedings of the 2nd European Conference on Mobile Robots (ECMR'05)*, pages 152-157, Ancona, Italy, September 2005.
5. K.Heero, A.Aabloo, M.Kruusmaa. Learning Innovative Routes for Mobile Robots in Dynamic Partially Unknown Environments. *International Journal of Advanced Robotic Systems*, 2(3):209-222, 2005.

ABSTRACT

This thesis investigates path planning strategies for mobile robots in large partially unknown dynamic environments. The aim of this work is to reduce collision risk and time of path following in cases when robot repeatedly traverses between predefined target points (e.g. transportation or surveillance tasks). A novel path selection strategy is examined. The method creates innovative paths between pre-defined target points and learns to use paths that are more reliable. This approach is implemented on the research robot Khepera and verified against the shortest path following by a wave transform algorithm. Experimental data show that new approach is able to reduce collision risk, travel time and distance. The robot is also able to learn and adapt quickly in a changing environment. Test results show that trajectory planned by a wave transform algorithm is very difficult to predict and control, because even little unmodelled obstacles can cause a large deviation from the pre-planned path. The approach used in this thesis makes robot motion more predictable. This thesis also suggests that the behaviour of the robot depends strongly on the knowledge about it's surrounding but not on the path planning strategy used. It is concluded that in order to optimise travel time, distance and deviation one has to minimize the occurrence of unknown obstacles since the last one influences the former parameters. Finally, this thesis addresses the problem of the utility of exploration on time-critical mobile robot missions. It is argued that in large environments mission-oriented mobile robot applications can become more efficient if the exploration strategy considers knowledge already gained and its applicability during the rest of the mission.

1 INTRODUCTION

Robotics is the science and technology of robots, their design, manufacture, and application. There are large areas needing further research: robot mapping, scalable architectures, planning, and world modelling, etc [36, 38].

Navigation is a critical ability for robots that claim to be mobile. It encompasses the ability of the robot to act based on its knowledge and sensor values so that it could reach its goal position as efficiently and reliably as possible. Navigation involves sensing, acting, planning, architecture, hardware, computational and power efficiencies, etc.

Planning is one obvious aspect of navigation that answers the question: what is the best way there? Given a map and a goal location, path planning involves identifying a trajectory that will cause the robot to reach the goal location when executed. Path planning is a strategic problem-solving competence, as the robot must decide what to do over the long term to achieve its goals.

This thesis contributes to the area of mobile robotics and path planning in dynamic partially unknown environments based on the articles added to the Appendix.

1.1 Motivation

Many mobile robot applications assume repeated traversal between predefined target points. For example, a mobile robot can be used in industry to transport details between a store and an assembly line. In military applications the ammunition transportation is quite usual. Possible working environments are also harbours, airports, landfills, etc. Also, a mobile robot could be used for surveillance what implies visiting certain checkpoints on a closed territory. There are lots of different scenarios where this kind of mobile robot repeated traversal is needed.

Real-word environments for this kind of mobile robot applications are complex, large, often unstructured and dynamic by nature. The robot has to navigate around obstacles that can have an arbitrary size, shape, location, and appear or disappear after an unknown time period. However, obstacle avoidance, no matter how good it is, always implies a collision risk due to the uncertainty in sensor information and motion planning, localization errors, terrain inequalities or computational imprecision. The robot may harm itself, slipping into the hole, or getting stuck in jutted constructions, etc. Uncertainty in the environment can always be hazardous for a mobile robot.

At the same time, mobile robots are expected to be efficient in a sense of their energy and time consumption. They have to fulfil their mission as fast and

safety as possible. Furthermore, in human inhabited environments safety of humans and therefore the reliability of the robot are primary.

To navigate successfully, the robot has to acquire a model of the environment where it is operating. Unfortunately it is impossible to keep the world model up to date and in accordance with all changes. The robot has limited recourses and has to struggle with uncertainty caused by changing and large (partially) unknown environment. On the other hand, the robot has a mission to complete and therefore it cannot spend all recourses just for exploration and world model updating (if the mission itself is not exploration of the environment). In spite of difficulties the robot has to be able to adapt with changes of the environment with the minimal effort. All this implies application of learning strategies that are strongly oriented to the mission, since the ultimate goal is to make the robot to do the right thing.

Path planners used in robotics have been proven to give globally optimal routes in globally known static environments. The optimality is usually measured in terms of distance. Other measures are also used, e.g. planetary rovers consider roughness and slope of the terrain to find reliable paths [15, 18]. However, their efficiency in complex, dynamic and partially unknown environments during long periods of time has not been investigated. Very few research studies reported so far consider the problem of path selection in changing environments. Approaches [16, 17, 19] assume that the structure of the environment is known *a priory*. In [29] it is assumed that unknown environment is static and does not change over time. In [24] uncertainty without two latest assumptions is considered.

1.2 Problem Statement

The general problem this thesis aims at solving is to find reliable paths for repeated traversal between previously determined target points so that following them minimises collision risk, speeds up the mission and increases predictability of the robot's behaviour.

Following presumptions are made to specify the problem.

1.2.1 Presumptions

It is assumed that the environment is dynamic and large. The structure of the environment is unknown and there exist obstacles with unknown size, orientation and location.

Mapping, path planning and localisation are not the main objectives of the robot. These are presumptions to make the successful completion of the actual mission possible. The robot is expected to fulfil its mission as fast and safely as possible.

Sensorial capabilities of the robot are insufficient to distinguish between static, dynamic and semi-dynamic obstacles.

Location errors are small and do not accumulate. This enables to follow a pre-planned path rather precisely.

Consequently it is not feasible to model the world precisely and/or keep it constantly updated.

1.3 Contribution of the Thesis

This thesis studies path selection strategies in complex dynamic and partially unknown environments during long periods of time. The contributions is a novel path-planning and selection strategy as well as general conclusions about path planning strategies based on experimental data.

One of the conclusions suggests that suboptimal paths (generated with algorithm in Section 4.1) are at least as good as shortest paths. Also a decision-making strategy is proposed to decrease collision risk and speed up the mission if the robot traverses repeatedly between predefined target points in dynamic partially unknown environments.

Additionally, time-critical mission-oriented exploration heuristics is presented for mobile robot applications were the robot is operating in large hazardous and unknown environments.

2 NAVIGATION IN DYNAMIC PARTIALLY UNKNOWN ENVIRONMENTS

Mobile robot path planning is typically stated as getting from one place to another. The robot must successfully navigate around obstacles to the target point and do it efficiently. The Holy Grail is to find the best route to the goal. The choice of an appropriate path planning method strongly depends on the model of the world.

Most of real life environments are complex and for path planning in such environments the key issue is how to model that complicated environment. However, the nature of the robot's operation and as well the precision with which the robot needs to achieve its goal determines the method. The mapping of the surrounding can be the main purpose of the robot e.g. exploration of the planetary rovers [49]. Reactive robots based on behaviours do not need a map at all [31]. But we cannot avoid a world model if we expect deliberative behaviour of the robot, like planning.

2.1 World Models and Path Planning

World models are usually divided roughly into two categories: topological (route, qualitative) and metric (layout, grid-based) models [43]. At present, lots of researches have produced a great variety of path planning methods [31, 37].

2.1.1 Topological Maps

Topological maps describe the connectivity of specific places called landmarks or gateways. These maps are represented in a form of a graph, where nodes are distinct places and edges are connections between them. The value of an edge may reflect then traversability of the respective segment of the path. Additional information may be attached to edges, such as direction, approximate distance, or the behaviours needed to navigate that path. If the robot finds a landmark and it appears on a map, the robot is localized with respect to the map. Examples of topological approaches include the works [16, 32, 44].

Widely used generalized Voronoi diagrams (GVD) also fall into this category [7]. GVD is a mapping method that tends to minimize the distance between the robot and obstacles on the map. The diagram consists of the lines constructed from all points that are equidistant from two or more obstacles in the plane. Hierarchical generalized Voronoi graphs (HGVG) is a roadmap that is an extension of GVD into higher dimensions then two. Choset [8] introduces

technique to incrementally construct the HGVG as the robot explores its unknown static environment using only line of sight information.

2.1.1.1 Path Planning on Topological Maps

Paths can be computed between to points using standard graph algorithms, such as the classical Dijkstra's single source shortest path algorithm [10]. The Voronoi diagram has a significant weakness in the case of limited range localisation sensors, since path planning algorithm maximises the distance between the robot and objects in the environment.

However, the number of paths from one place to another is limited by the number of edge combinations.

2.1.2 Metric Maps

Metric maps capture geometric properties of the environment. The number of different map representations is very large; none of them is dominant. The most common ones are regular grids and quadtrees (and their 3D extension, octrees).

Regular grid is a two-dimensional array of square elements (called pixels). Regular grids are often called as occupancy grid, because each element in the grid will hold a value representing whether the location in space is occupied or empty [11]. Unfortunately, regular grids do not scale up very well. The size of the map grows with the size of the environment and path planning becomes computationally expensive. On a coarse grid, path planning is faster but obstacles are expanded on the grid and narrow corridors can disappear. One commercial robot that uses a standard occupancy grid is the Cye robot [2]. Also the tour-guide robots Minerva [42] and Rhino [4] are using occupancy grids.

Quadtrees are recursive grids. They are created by recursively subdividing each map square with non-uniform attributes into four equal-sized sub-squares. The division is repeated until a square is uniform or the highest resolution is reached. Quadtrees reduce memory requirements hereby allowing efficient partitioning of the environment. A single cell can be used to encode a large empty region [48].

However, the distinction between metric and topological maps has always been fuzzy, since virtually all topological approaches rely on geometric information. In practice, metric maps are finer grained than topological ones.

2.1.2.1 Path Planning on Metric Maps

Most world representation can be converted to graphs (e.g. cell decompositions, 4-connected and 8-connected grids, etc.). Typically graph-based path planners rely on A* or D* algorithms [6, 39] or on their modification (Incremental A* [21], Focussed D* [40], D* Lite [20], Delayed D* [13]). These algorithms

generate shortest paths reducing computational complexity in case of highly connected graphs such as regular grids.

A* algorithm finds a path as good as found by Dijkstra's algorithm but does it much more efficiently using an additional heuristic to guide itself to the goal. Dijkstra's algorithm uses a best first approach. It works by visiting nodes in the graph starting from the start point and repeatedly examining the closest not-yet-examined node until it reaches the goal. A* always first expands the node with the best cost calculated by $f(n) = g(n) + h(n)$. Where $g(n)$ represents the cost of the path from the starting point to the node n, and $h(n)$ represents the heuristic estimated cost from the node n to the goal. Usually, for calculating the heuristic cost, the Manhattan or the Euclidean distance is used.

D* is the dynamic version of A* producing the same result but much faster in dynamic environments. In a sense of replanning A* is computationally expensive because it must replan the entire path to the goal every time new information is added. In contrast, D* does not require complete replanning since it adjusts optimal path costs by increasing and lowering the cost only locally and incrementally as needed. Expansions of D* algorithm, like Focussed D*, D* Lite, Delayed D*, are accordingly even more efficient.

Potential fields planners are very widely represented, since they are extremely easy to implement. The potential field method treats the robot as a point under the influence of an artificial potential field. The goal acts as an attractive force on the robot and the obstacles act as repulsive forces. Such an artificial potential field smoothly guides the robot to the goal while simultaneously avoiding known obstacles. While potential field planners follow the gradient descent of the field to the goal they always find the shortest path from every possible start point. Potential fields have become a common tool in mobile robot application in spite of the local minima problem [6]. Harmonic functions can be used to advantage for potential field path planning, since they do not exhibit spurious local minima [9].

A popular family of path planning methods on grids is wavefront-based planners. They are based on potential fields, but do not have local minima problem [2]. The basic principle is that the configuration space is considered to be a conductive material with heat radiating out from the initial node to the goal node. Finally the heat will spread and reach the goal, if there is a way. The optimal path from all grid elements to the goal can be computed as a side effect. The distance transform planners are well-known wavefront-based planners propagating distance throughout each grid cell in an outward direction from the specified goal point to the start point filling the entire free space. The optimal path from all grid elements to the goal is then found by using the steepest descent trajectory. Zelinsky introduced a safe path transform method in [50]. In addition to propagating a distance wavefront from the goal, another wavefront is propagated which is a combination of the distance from the goal together with a measure of the discomfort of moving near obstacles. In [45] distance transform is extended with linear vector combination to estimate shortest global path and

obtain the safe local direction in which a mobile robot moves safely in a local environment. Trulla also is one of the many wavefront types of path planners [30]. This algorithm initially computes all possible paths from all possible locations to the goal. Trulla output is a potential field-like representation of the best direction the robot should take from any location in the map to the goal, given the *a priori* map and terrain preferences.

2.1.3 Hybrid Approaches

Frequently, metric and topological methods are used together. Those hybrid methods try to combine the advantages of metric-based and topological planning approaches, since both paradigms have strengths and weaknesses.

For example, topological approaches often have a difficulty determining distinct places if they look alike. This can be caused by sensor noise and aliasing. Also, since sensory input usually depends strongly on the viewpoint of the robot, it may fail to recognize geometrically nearby places even in static environments. All this makes construction and maintenance of large-scale maps difficult, particularly if sensor information is highly ambiguous. The key advantage of topological representation is their compactness, what is the main shortcoming of the metric maps. Due to compactness, topological representation permits faster planning than the metric approach. On the other hand, metric maps permit much more detailed path planning due to the high resolution. Since topological approach usually does not require the exact determination of the geometrical position of the robot, it often recovers better from drift and slippage-phenomena that must constantly be monitored and compensated on metric-based approaches.

Byun and Kuipers [27] used a multi-level spatial hierarchy. The lowest level is identifying landmarks. The next layer up is topological, represented on a relational graph, which supports planning and reasoning. The uppermost level is metric, where the agent learns the distances and orientation between the landmarks and can place them in fixed coordinate system. Fabrizzi and Saffiotti [12] extract the topological map from the previously created grid map analysing the shapes of the free spaces. Thrun [41] generates topological maps on top of the grid-based maps by partitioning the latter into coherent regions, separated by critical lines. Critical lines correspond to narrow passages such as doorways. By partitioning the metric map into a small number of regions, the number of topological entities is several orders of magnitude smaller than the number of cells in the grid representation. Poncela *et al.* [33] perform exploration path planning on two levels: global planning is performed on topological level and local planning is performed on metric level. Such representation permits exploration in a fast and efficient way. Kruusmaa [22, 23] uses case-base reasoning with a grid map. A grid-based map permits detailed path planning and case-base stores travelled paths with traversability information of those paths in a form of a simple cost function that is easy to update.

2.2 Representing Uncertainty

To carry out complex missions in unknown or partially unknown environments, the robot must be able to incrementally generate and maintain a map of this environment. Usually, it gathers sensor information to update its word model during the traversal [43]. Mapping problem often occurs in conjunction with the localization problem. To estimate where things are in the environment and determine the pose of the robot needs to be solved concurrently. This is often called the simultaneous localization and mapping (SLAM) problem [7].

Usually physical environments change over time. From the viewpoint of the robot it means appearance and disappearance of obstacles in arbitrary places and time, or in the worst case, change of the whole structure of the environment. Dynamic objects are usually considered to be moving obstacles like people, cars, strollers, etc. But there exists another class of dynamic obstacles with much more discrete motion, for example objects stored on the factory floors and warehouses, lightweight furniture, details on construction sites, etc.

In all such dynamic environments mapping is a big challenge, since even mapping of a static environment is hard problem due to sensor noise, localization errors and imprecise motion control. To acquire global information, the robot has to actively explore its environment. Therefore the precision of the word model depends on the region size and on the intensity of survey. However, in a large dynamic environment sooner or later the world model will be desperately incorrect. Due to complexity and dynamism, it is principally impossible to maintain exact models and to predict their accuracy.

Vast majority of published algorithms make a static world assumption, and hence are principally unable to cope with dynamic environments [43]. Instead, the predominant paradigm relies on a static word assumption, in which the robot is the only time-variant quantity (and everything else that moves is just noise). The problem of dynamic obstacles is usually tackled in the context of collision avoidance [14, 34, 47].

There are some attempts to model dynamism, but this field is poorly explored. Biswas and *et al.* [3] have proposed an occupancy grid-mapping algorithm ROMA (robot object mapping algorithm) capable of modelling non-stationary environments. Their approach uses a straightforward map differencing technique to detect changes in an environment over time. By combining data from multiple maps while learning objects models, the resulting models have higher fidelity than could be obtained from any single map.

A robot using uncertain and inaccurate metric maps can miss short and easily traversable paths. Therefore the shortest (optimal) path-planners cannot demonstrate their advantage like in the completely known environments, since replanning is unavoidable. In such dynamic environments the best path to the goal is not necessarily the shortest. Taking a longer path can sometimes reduce the collision risk and speed up the mission.

A topological word-model is much easier to update, since the robot does not have to know where the obstacles exactly lay and of what shape and size they are. It only matters how safely and fast the robot follows its path to the goal. But in dynamic environment it is quite difficult to guarantee that robot can easily determine new landmarks or distinguishable places that do not change their location or disappear at all. The robot may get confused and lost.

Storing travelled paths with traversal information is even more flexible then a topological map. The memory does not have to be reorganized when the environment changes. It also permits storing a more detailed description of paths and discovering more edges.

3 REPEATED TRAVELLING

The necessity of exploration in unknown or partially unknown environments depends on the nature of the robot's mission. When the entire environment should be mapped out or covered (e.g. de-mining, search tasks), full coverage algorithms are used [5, 33, 49]. To reach a specific target location only once, the navigation algorithms are used [21, 39, 50]. The cases when the robot has to travel repeatedly a long period of time between predefined goal points in an unknown dynamic environments is the combination of both algorithms. The robot needs to find the right balance between exploration of the environment and performing the actual task via a known suboptimal path. During the mission the environment may change and the robot has to adapt to changes. Typical navigation algorithms try to find optimal (shortest) paths to the goal. In such complex environments the best path to the goal is not necessary the shortest. Depending on the nature of the environment, there may exist routs that are longer but easier to follow.

4 THE APPROACH

The contribution of this thesis is a new approach to repeated traversal under uncertainty. This approach is based on a hybrid path planning method represented in [23]. This is a path planning approach conducted on grid map, which permits detailed planning and finds many alternatives for a path planning problem. Instead of modelling an environment the traversed paths are stored in robot's memory. By remembering the past routes the robot learns about the environment and uses this information to choose the paths, which are most likely to be easily traversable. The grid map is not updated at all, since the robot cannot decide which obstacles will be removed and which will stay for a longer period. The map will just describe start and goal points and the general geometry of the environment. If necessary, static obstacles can be stored on the map by an operator (e.g. walls, very hazardous areas). The exploration of the environment is thus conducted in conjunction with repeated traversal. To explore the environment innovative paths are generated using probabilistic path transform method that propagates cell values along the map. Each cell, which does not contain a fixed obstacle, gets a value, which is a combination of the distance from the goal, the measure of the discomfort from moving near obstacles and a parameter with a random value. This method is a modification of the path transform algorithm [50].

This work improves and extends the work represented in [23]. The drawback of [23] probabilistic path generation algorithm is that there is no guarantee that the generated path is different from the original wave transform algorithm and that the algorithm will find all possible paths from start to goal. This thesis aims at improving innovative path generation algorithm by eliminating previously noticed disadvantages. For that we have to establish following requirements to the path selection algorithm:

- Paths are as much as possible different from each other to let the robot to find out as many innovative solutions as possible.
- The algorithm is able to discover all virtually possible alternatives.
- The algorithm covers the whole space of innovative paths with as few alternatives as possible to maintain the robot's ability to generalize and keep the memory constrained.
- All paths are easy to follow if free from obstacles.

A novel path selection algorithm (described in next Section 4.1), which satisfies the criteria above, is used for this purpose.

4.1 Novel Path Generation Algorithm

The path generation algorithm described in [46] is based on covering the working area with paths segments. The whole area is divided into small path fragments (of length 2, shown in Figure 1) and the generated paths cover all these segments. The paths are limited to have only right and up moves. It will exclude all the unnecessarily long and complex paths (actually all paths having back turns).

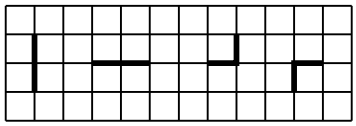


Figure 1. Path fragments of length 2

This algorithm gives a relatively small number of different paths of the minimal cover and scales up very well. It is proven that for a grid of the size $m \times n$ the cardinality of the minimal cover is $2m + 2n - 2$ [46]. It grows linearly with a small constant. Figure 2 illustrates one possible cover of the 3×4 grid.

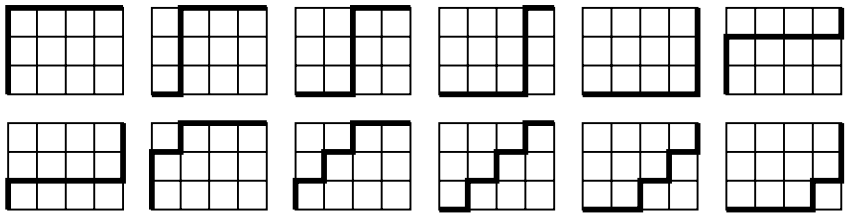


Figure 2. Cover of the 3×4 grid

The drawback of this type of paths is that the robot would not operate efficiently in a maze-like environment. However, most of real environments are not mazes. The second shortcoming is the occurrence of the zigzagged paths since it is only allowed to move right and up. This is typical to all grid-based path planners and mobile robots usually use path relaxation techniques to smoothen the path at runtime.

While [46] gives a through insight to the theoretical aspects of the path generation algorithm, this thesis investigates its advantages in practice (experimental design is described in Chapter 5).

4.2 Exploration, Learning, and Decision-making

Instead of modelling the environment as accurately as possible paths where inaccuracy of the world model does not significantly influence the result of path planning are found. A set of suboptimal paths is chosen and their traversability evaluated by trial and error until the satisfied criteria of safety and reliability is found. The general form of the path selection algorithm is represented in Figure 3.

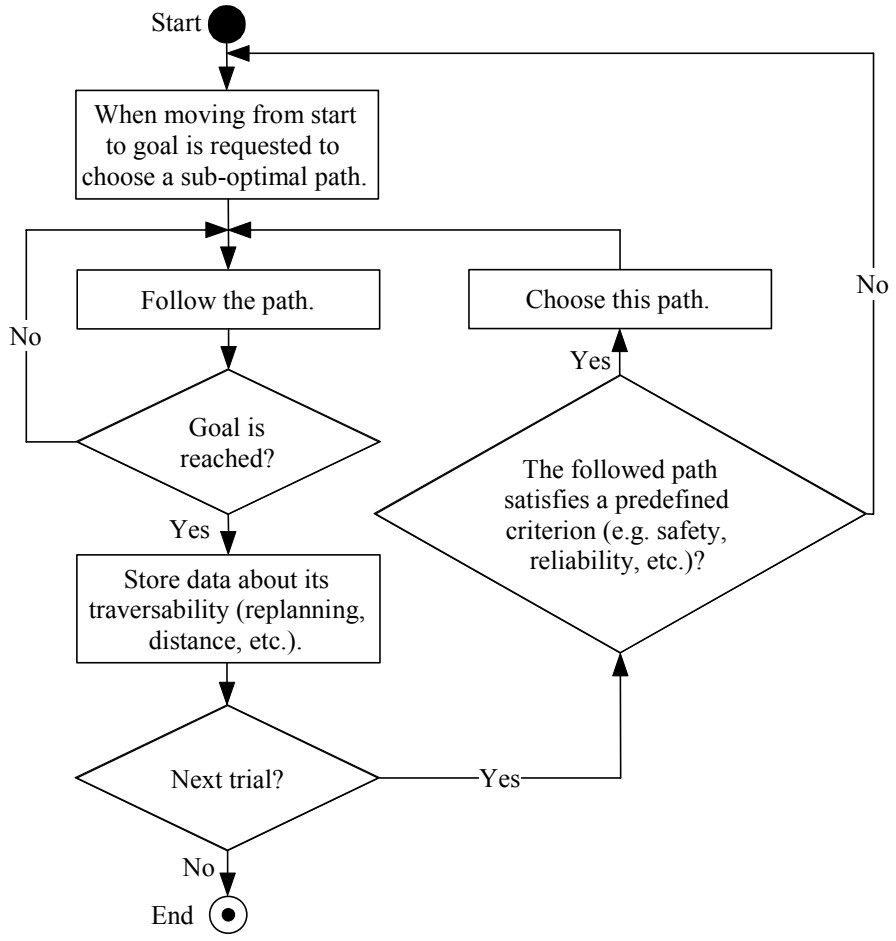


Figure 3. The general path selection algorithm

An algorithm described in Section 4.1 is used for path generation. The followed path usually differs from the pre-planned one because unexpected obstacles deviate the robot from its initial course. It is also crooked since the robot repeatedly corrects its localization errors and avoids collisions with obstacles. Therefore the zigzags of the followed path are straightened and gaps and cycles are removed before storing it. This path relaxation is documented in [25].

For global (and local) replanning a wave transform algorithm is used [50]. If an obstacle has blocked the previously planned path, then this path is abandoned and the shortest path to the goal is generated. Since the environment is changing and word-model is more or less imprecise, the algorithm does not try to compile paths from parts previously known to be good. Rather it tries to generate and evaluate the path as a whole.

After reaching the goal the traversed path is stored with its statistical data: number of replannings, travelled time, travelled distance, and deviation form the originally pre-planned path. This approach tries to minimize the hazard of path following. Therefore paths with lower number of replannings are preferred. The criterion of path selection among the stored path is the probability inversely proportional to the number of unexpected obstacles.

At the next trial the stored paths are examined to determine whether some of them satisfy the criterion. If such a path exists, this path is followed. Otherwise a new suboptimal path is generated and followed.

The experimental results show that such an exploration, planning and decision-making procedure minimises risk, time of path following and increases the predictability of robot's behaviour.

The pseudo code of the described learning method is represented in Appendix F.

4.3 Limits

This approach has some severe limits. The method assumes that the robot will traverse repeatedly between predefined target points. Only this restriction makes it possible to learn to use the most reliable paths by trial and error.

Second, it assumes that localisation errors are small and do not accumulate allowing the robot to follow planned trajectories fairly precisely. One possible solution is to use this method with global positioning system (GPS), differential global positioning system (DGPS) or with pseudolite navigation [1, 28].

Third, the robot may stuck in a local minimum and not able to test a path away from its current selection. Whereas the environment is dynamic, the movement of the obstacles will deflect the robot away from the current route after some period of time. Also, we have no guarantee in an unknown environment that somewhere exists a better route and it is easy to find. This assumes full exploration of the environment.

5 EXPERIMENTAL DESIGN

All experiments are conducted using a well-known research robot Khepera (shown in Figure 4) manufactured by K-Team [26]. It is a circular mini-robot (diameter is 53 mm) with differential drive and with eight on-board infrared range sensors for collision avoidance. It can be connected and commanded with PC over a serial link. Due to Khepera's small size it is much easier to create large environments with respect to the size of the robot and control the changes in this environment to validate the algorithms.

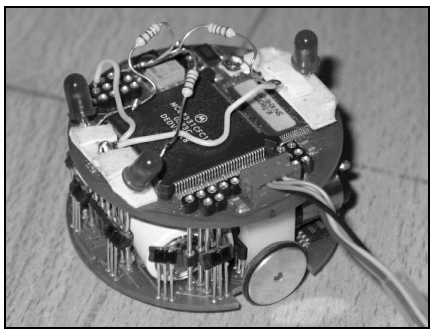


Figure 4. Research robot Khepera

The behaviour of the robot is controlled with a PC program written in C++ [35, Appendix G]. It has the GUI to create *a priori* map with known obstacles, to allocate mission target points, to plot difficult areas which slow down the motion of the robot, to track the passed paths of the robot, etc. All the interesting statistics about the experiments is saved into the file for later analysis.

Since Khepera lacks sensors for accurate localization and there is considerable error due to the contact between the wheels and the surface or dust inside the motors the localization is implemented using a global vision system. A video camera is mounted to the ceiling above the test environment to recognize the pose of the robot. Additionally, 3 LED in a form of an isosceles triangle are mounted on the robot to make robot's pose recognition easier. During every localisation episode the current image of the camera is processed in the host computer to identify the robot's position and orientation (shown in Figure 5). The vision system is used only to update the location of the robot. Since the robot does not need to localize itself by means of odometry, landmarks or onboard sensors but uses an external system, the localization errors are rather small (comparable to the size of the robot) and do not accumulate.

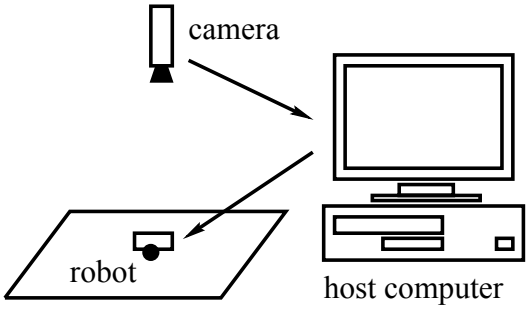


Figure 5. Localization system

The size of the test environment varies from 1860×1390 up to 2320×1710 mm and the surface is flat. With respect to the robot's size and the range of the infrared sensors the environment can be considered to be rather large.

In the dynamic environments all dynamic obstacles are "movable" (replaced after every traversal of the robot from the start point to the goal point) but not moving as usually is considered. This enables exactly control the environment and interpret experimental results. It is very difficult to control the real environment with moving obstacles.

All test environments in detail are represented in Appendix E.

6 INTRODUCTION TO CONTRIBUTING PUBLICATIONS

This chapter gives a brief introduction to the publications that contribute to this thesis. All these publications are given in full in the appendices A, B, C and D.

6.1 Path Selection for Mobile Robots in Dynamic Environments (Paper Introduction)

This is the first paper that investigates the performance of the path generation algorithm (described in Section 4.1) on the real robot. Verification of the path selection algorithm's performance has conducted in the dynamic and totally unknown environment.

The wave transform algorithm to generate shortest paths is compared to path selection algorithm with global replanning. All four measured parameters (replanning count, traversal time, deviation, distance) showed better results using path selection algorithm. First experiments confirmed the usability of the approach if even very little is known about the surrounding or when the environment is completely restructured during the mission.

6.2 Robots Find a Better Way: A Learning Method for Mobile Robot Navigation in Partially Unknown Environments (Paper Introduction)

This paper presents results of the second series of experiments of the path selection algorithm focused on partially known environments. The hypothesis was that as soon as the environment becomes better known to the robot, the shortest path following strategy would outperform the investigated approach.

The test results did not confirm that initial hypothesis. On the contrary, it appeared that even if small obstacles are unmodelled, the path selection algorithm improves the performance. The same comparison between path planning strategies has conducted as in the first paper. Test environments were kept static to find out the relation between the environment model and the behaviour of the robot. Three test environments were examined: all obstacles were modelled, only large obstacles were modelled and only small obstacles were modelled on *a priory* map.

Operating in the totally known environment the robot is able to use all information available and plan the globally best paths. Statistical data showed that the behaviour of the robot was well predictable and stable. The deviation

from the original path was not more then the diameter of the robot and only once a detected obstacle or sensor noise forces the robot to replan its path. Therefore, it can be concluded that localisation errors, imprecision of mechanical linkages or the control program did not influence the test results significantly.

As soon as the environmental model becomes partially unknown the trajectory of the robot was very difficult to predict and control. Even small obstacles could cause large deviation from the pre-planned path. More over, it can be concluded that optimal (shortest) path planning is not a relevant problem in the partially unknown environments. As soon as the robot does not have all global knowledge available, suboptimal solutions give at least as good results as the optimal one. This implies that much more importance should be paid on modelling the environment and its changes.

6.3 Learning Innovative Routes for Mobile Robots in Dynamic Partially Unknown Environments (Paper Introduction)

This paper presents an additional investigation of the path selection algorithm with local replanning and it's comparison with all previously conducted experiments. When unexpected obstacle is detected, local replanning tries to find its way back to the pre-planned path avoiding sub-goal obsession [31].

It is concluded that global replanning does not improve the performance compared to the local replanning, since the experimental results do not reveal any significant difference in performance. The efficiency of path planning rather depends on the world model than on the planning strategy. A global planner that does not have all global information available anyway fails to make a globally optimal plan and therefore the locally replanning agent performs equally well.

The overall results show that in a complex environment there may exist paths that are easier to follow then the shortest paths. Finding and following them helps to reduce collision risk as well as to minimize travel time, distance and deviation from the originally planned path. It appears that in all different environments travel time and the number of replannings are highly correlated.

6.4 On the Utility of Exploration on Time-Critical Mobile Robots Missions (Paper Introduction)

In an unknown or partially unknown environment the robot learns mainly by trial and error. In hazardous environments learning and exploration can lead to undesirable damages. If the robot in addition has a time-critical mission, the

utility of expensive exploration becomes questionable. From the utilitarian point of view, knowledge is useful only as long as it increases the performance of the robot and thus helps to fulfil the mission. The knowledge that can be used many times is much more valuable than knowledge used only once.

This paper considers mission-oriented exploration heuristics for mobile robot applications based on the above statements. It proposes a heuristic strategy that chooses between exploring new areas and exploiting knowledge about the already explored areas. The robot having a mission plan, considers the amount of knowledge acquired so far and its applicability during the rest of the mission.

The mission plan of the robot in a large unknown environment consists of target points that it has to reach in a predefined order. Every traversal between two target points can be viewed as a task of the mission. Usually exists number of similar tasks (e.g. repeated traversal between some target points) in the mission. The heuristics of the decision maker at the current task tends to explore when similar task is not encountered often in the past and if it is needed often during the rest of the mission. The sooner during the mission new knowledge will be needed the more exploration is preferred.

This strategy is verified against greedy one that always chooses a new path, if it is predicted to be better than the best path known so far. Experimental results show that the robot in the test environment using the heuristic strategy fulfils the mission faster than robot using the greedy strategy.

CONCLUSIONS

This thesis investigates path planning strategies for repeated traversal in large dynamic partially unknown environments. The aim of the approach was to minimize collision risk and speed up the mission by adapting to the changes in the dynamic environment.

The advantages of the novel path selection algorithm for generating innovative paths between predefined target points are demonstrated. Over 600 test runs are conducted using the research robot Khepera (all descriptions of the experiments and experimental results are represented additionally in Appendix G). The behaviour of the robot is verified against the shortest path following strategy in various complex environments (varying from static to dynamic as well as from unknown to partially and totally known).

The experimental results lead to the following general conclusions.

Path planning approach presented in this thesis can be used even if very little is known about the environment or when the environment is completely restructured during the mission. The path selection algorithm will efficiently cover the whole space even if the environment is large. This approach helps to reduce time, risk of collisions and increases the predictability of robot's behaviour.

To optimize travel time, distance, energy consumption, collision risk or deviation from the original path, unexpected events should be decreased as changes in the former parameters depend on the last one.

In an uncertain environment the trajectory of the robot is very difficult to predict and control because the deviation from the planned path is weakly correlated to the accuracy of the world-model.

Optimal (shortest) path planning is not a relevant problem in partially unknown environments. The behaviour of the robot is influenced by the knowledge it has about the environment but does not depend on the path planning strategy. In order to increase the reliability of mobile robot applications, much more attention should be paid on modelling the environment and its changes than an optimisation of path planning algorithms.

Gaining as accurate as possible knowledge about the surrounding is not necessary beneficial if the mission time is limited in a large hazardous environment. Mission-oriented exploration heuristics could be considered in mobile robot applications that are time-critical, where the robot is operating in a large unknown environment and when this environment is dangerous.

Obviously these experimental results cannot be interpreted as applying to all possible environments because in appendices A, B, C only one randomly generated base environment was used as well as in the study in Appendix D. But if these constructed test worlds are good enough approximations of a large, unstructured, uncertain and dynamic environment, these results could be

generalised to other cases. The influence of the moving obstacles on the results is unknown because these type of obstacles are avoided to ensure exact control over the test environment. However, this affects only replanning algorithm and not global path selection.

These conclusions certainly cannot be generalised to topological planning since the models and algorithms considerably differ from those used for grid maps.

REFERENCES

1. V.Ashkenazi, D.Park, M.Dumville. Robot Positioning and the Global Navigation Satellite System, *Industrial Robots: An International Journal*, 27(6):419-426, 2000.
2. P.H.Batavia, I.Nourbakhsh. Path planning for the Cye personal robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.
3. R.Biswas, B.Limketaki, S.Sanner, S.Thrun. Towards Object Mapping in Dynamic Environments with Mobile Robots. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.
4. J.Buhmann, W.Burgard, A.B.Cremers, D.Fox, T.Hofmann, F.Schneider, J.Strikos, S.Thrun. The Mobile Robot Rhino. *AI Magazine*, 16(1), 1995.
5. H.Choset. Coverage of Known Spaces: The Boustrophedon Cellular Decomposition. *Autonomous Robots*, 9:247-253, Kluwer, 2000.
6. H.Choset, K.M.Lynch, S.Hutchinson, G.Kantor, W.Burgard, L.E.Kavraki, S.Thrun. *Principles of Robot Motion*, The MIT Press, 2005.
7. H.Choset, K.Nagatani. Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization. In *IEEE Transactions on Robotics and Automation*, 17(2):125-137, April 2001.
8. H.Choset. Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph. *PhD thesis*, California Institute of Technology, 1996.
9. C.I.Connolly, R.A.Grupen. The Application of Harmonic Functions to Robotics, *Journal of Robotic Systems*, 10(7):931-946, 1992.
10. T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein. *Introduction to Algorithms*, Section 24.3: Dijkstra's algorithm, pages 595-601, Second Edition, MIT Press and McGraw-Hill, 2001.
11. A.Elfes. Using occupancy grids for mobile robot perception and navigation, *IEEE Computer*, 22(6):46-57, 1989.
12. E.Fabrizi, A.Saffiotti. Extracting Topology-Based Maps from Gridmaps. In *Proceedings of the 2000 IEEE International Conference of Robotics and Automation (ICRA 2000)*, pages 2973-2978, 2000.

13. D.Ferguson, A.Stentz, The Delayed D* Algorithm for Efficient Path Replanning. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, April 2005.

14. P.Fiorini, Z.Shiller. Motion Planning in Dynamic Environments. *The 7th International Symposium of Robotics Research*, pages 237-248, Munich, Germany, October 1995.

15. D.B.Gennery. Traversability Analysis and Path Planning for Planetary Rovers. *Autonomous Robots*, 6:131-146, Kluwer, 1999.

16. K.Z.Haigh, M.M.Veloso. Planning, Execution and Learning in a Robotic Agent. *The 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98)*, pages 120-127, June 1998.

17. K.Z.Haigh, M.M.Veloso. Route Planning by Analog. In *Proceedings of Case-Based Reasoning Research and Development, First International Conference (ICCBR-95)*, pages 169-180, Springer-Verlag, 1995.

18. A.Howard, H.Seraji. Vision-Based Terrain Characterization and Traversability Assessment. *Journal of Robotic Systems*, 18(10):577-587, Wiley periodicals, 2001.

19. H.Hu, M.Brady. Dynamic Global Path Planning with Uncertainty for Mobile Robots in Manufacturing. *IEEE Transactions on Robotic and Automation*, 13(5):760-767, October 1997.

20. S.Koenig, M.Likhachev. Improved Fast Replanning for Robot Navigation in Unknown Terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

21. S.Koenig, M.Likhachev. Incremental A*. In *Proceedings of the Neural Information Processing Systems*, 2001.

22. M.Kruusmaa. Global Level Path Planning for Mobile Robots in Dynamic Environments. *Journal of Intelligent and Robotic Systems,* special issue for Robot Motion Planning, Kluwer, 38(1):55-83, September 2003.

23. M.Kruusmaa. Global Navigation in Dynamic Environments Using Case-Based Reasoning. *Autonomous Robots*, Kluwer, 14(1):71-91, 2003.

24. M.Kruusmaa. Repeated Path Planning for Mobile Robots in Dynamic Environments. *PhD thesis*, Chalmers University of Technology, Gothenburg, Sweden, 2002.

25. M.Kruusmaa. Repeated Path Planning for Mobile Robots in Uncertain Environments. In *Proceedings of the IASTED International Conference on Robotics and Automation*, pages 226-231, 2001.

26. K-Team official website. Available at *http://www.k-team.com*. Visited on 19.03.2006.

27. B.Kuipers, Y.T.Byun. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representation. *Robotics and Autonomous Systems*, 8: 47-63, 1991.

28. E.Lemaster, S.Rock. A Local-Area GPS Pseudolite-Based Navigation System for Mars Rover. *Autonomous Robots*, 14:209-224, 2003.

29. R.Meshulam, A.Felner, S.Kraus. Utility-based multi-agent system for performing repeated navigation tasks. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 887-894, The Netherlands, 2005.

30. R.R.Murphy, K.Hughes, A.Marzilli, E.Noll. Integrating explicit path planning with reactive control of mobile robots using Trulla. *Robotics and Autonomous Systems*, 27(4):225-245, Elsevier Science, 1997.

31. R.R.Murphy. *Introduction to AI Robotics*. The MIT Press, 2000.

32. U.Nehmzow, C.Owen. Robot Navigation in the Real World: Experiments with Manchester's Forty Two in Unmodified Large Environments. *Robotics and Autonomous Systems*, 33:223-242, Elsevier Science, 2000.

33. A.Poncela, E.J.Perez, A.Bandera, C.Urdiales, F.Sandoval. Efficient Integration of Metric and Topological Maps for Directed Exploration of Unknown Environments. *Robotics and Autonomous Systems*, 41:21-39, 2002.

34. E.Prassler, J.Scholz, P.Fiorini. A Robotic Wheelchair Roaming in a Railway Station, *IEEE International Conference on Field and Service Robotics*, Pittsburg, USA, August 1999.

35. Program code and experimental results of the thesis. Available at *http://math.ut.ee/~kristo/phd/*.

36. M.A.Salichs, L.Moreno. Navigation of Mobile Robots: Open Questions. *Robotica*, 18:227-234, Cambridge University Press, 2000.

37. R.Siegwart, I.R.Nourbakhsh. *Introduction to Autonomous Mobile Robots*, The MIT Press, 2004.

38. D.J.Spero. A Review of Outdoor Robotics Research. *Technical Report MECSE-17-2004*, Department of Electrical and Computer Systems, Monash University, Melbourne, November 2004.

39. A.Strentz. Optimal and Efficient Path Planning for Partially-Known Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1994.

40. A.Strentz. The Focussed D* Algorithm for Real-Time Replanning. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, 1995.

41. S.Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 18(1):21-71, 1998.

42. S.Thrun, M.Beetz, M.Bennewitz, W.Burgard, A.B.Cremers, F.Dellaert, D.Fox, D.Hähnel, C.Rosenberg, N.Roy, J.Schulte, D.Schulz. Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. *International Journal of Robotics Research*, 19(11):972-999, 2000.

43. S.Thrun. Robotics mapping: A Survey. *Technical Report CMU-CS-02-111*, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213, February 2002.

44. N.Tomatis, R.Philippsen B.Jensen, K.O.Arras, G.Terrien, R.Piguet, R.Siegwart. Building a Fully Autonomous Tour Guide Robot: Where Academic Research Meets Industry. In *Proceedings of the 33rd International Symposium on Robotics (ISR'2002)*, Stockholm, Sweden, October 2002.

45. S.Trihatmo, R.A.Jarvis. Short-Safe Compromise Path for Mobile Robot Navigation in a Dynamic Unknown Environment. Accepted for presentation at *Australian Conference on Robotics and Automation* 2003, Brisbane, December 2003.

46. J.Willemson, M.Kruusmaa. Algorithmic Generation of Path Fragment Covers for Mobile Robot Path Planning. In *Proceedings of the 3rd IEEE Conference on Intelligent Systems (IEEE IS'06)*, to appear.

47. F.Xu, H. van Brussel, M.Nuttin, R.Moreas. Concepts for Dynamic Obstacle Avoidance and Their Extended Application in Underground Navigation. *Robotics and Autonomous Systems*, 42(1):1-15, 2003.

48. A.Yahja, S.Singh, A.Stentz. An Efficient on-line Path Planner for Outdoor Mobile Robots. *Robotics and Autonomous Systems*, 32:129-143, Elsevier Science, 2000.

49. B.Yamauchi. A Frontier-Based Approach for Autonomous Exploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146-151, Monterey, July 1997.

50. A.Zelinsky. Using Path Transforms to Guide the Search for Findpath in 2D. *International Journal of Robotics Research*, 13(4):315-325, August 1994.

MOBIILSETE ROBOTITE TEE PLANEERIMINE JA ÕPISTRATEEGIAD DÜNAAMILISTES JA OSALISELT TUNDMATUTES KESKKONDADES

SISUKOKKUVÕTE

Mobiilsete robotite tähtsaimaks omaduseks on navigeerimisvõime. Edukaks navigeerimiseks vajatakse teede planeerimisalgoritme, mis võimaldavad robotil jõuda sihtmärgini efektiivselt ning takistustega kokku põrkamata. Tee planeerimine eeldab, et robotil on olemas ettekujutus keskkonnast, milles ta opereerib. Keskkonna mudeli loomine sõltub enamasti roboti andurite võimalustest tajuda ümbritsevat. Tee planeerimismeetodite valik baseerub omakorda sellel, kuidas reaalset keskkonda kujutatakse roboti mälus. Näiteks võrestikkaartidel on levinud lainefrondi planeerimisalgoritmid, graafina esitatud teede planeerimisalgoritmid põhinevad aga graafi lühimate või hinnanguliselt parimate teede otsimisel kahe tipu vahel.

Isegi staatilise keskkonna kaardistamine on oma olemuselt keeruline protsess, sest arvestada tuleb andurite müraga, vigadega roboti positsiooni määramisel, roboti liikumisest põhjustatud hälvetega, piiratud ressurssidega (näiteks arvutusvõimsus) ja muu taolisega. Seda enam on tee planeerimine komplitseeritum reaalsetes keskkondades, mis on enamasti oma olemuselt dünaamilised, kindla struktuurita või robotile osaliselt või täiesti tundmatud.

Paljud mobiilsete robotite rakendused eeldavad korduvat liikumist kahe või enama sihtmärgi vahel. Siia kuuluvad transpordiülesanded tööstus- ja militaarvaldkonnas, piirkonna seire, konvoeerimine, päästeoperatsioonid jne.

Käesolev väitekiri uuribki võimalusi, kuidas leida usaldusväärseid teid korduval liikumisel kahe eeldefineeritud asukoha vahel. Sealjuures tehakse järgmisi eeldusi:

- Keskkond on suur ja dünaamiline, selle struktuur pole teada ning suvaliste mõõtmetega takistused võivad asuda teadmata kohtades.
- Robotilt eeldatakse efektiivset ja turvalist missiooni täitmist, mis iseenesest ei ole keskkonna kaardistamine ja roboti positsiooni määramine. Viimased on vaid vajalikud meetmed eesmärgi saavutamiseks.
- Roboti andurid ei ole võimelised eristama dünaamilisi objekte staatilistest. Lokaliseerimishälbed on väiksed ning ei akumuleeru, mis omakorda võimaldab planeeritud teed suhteliselt täpselt järgida.

- Keskkonna keerukust ja roboti võimalusi arvestades ei ole võimalik keskkonna mudelit täpselt koostada ning tegelikkusega kooskõlas hoida.

Roboti võime adapteeruda ning käituda ennustatavalt keerukas keskkonnas eeldab selle tundmaõppimist. Käesolevas töös ei püüta selleks otseselt kaardistada tundmatut keskkonda vaid selle asemel hoitakse mälus juba läbitud teid koos statistiliste andmetega (läbitud tee pikkus, kulunud aeg, ümberplaneerimiste arv, jmt). Salvestatud teed on edaspidise efektiivsuse huvides teataval määral õgvendatud ning välja on visatud mõttetud tsüklid. Igal järgmisel katsel eeldefineeritud sihtpunktide vahel liikumiseks on robotil võimalus valida kas uue genereeritud tee või juba mällu salvestatud parima tee vahel. Tee planeerimisstrateegia püüab vähendada marsruudi läbimise ohtlikust valides sellise tee, mis on kõige tõenäolisemalt takistustest vaba. Valikukriteeriumiks juba salvestatud teede jaoks on tõenäosus, mis on proportsionaalselt pöördvõrdeline ümberplaneerimiste arvuga. Kui seatud kriteeriumile vastavat teed ei leita, siis robot kasutab sihtmärgini jõudmiseks uut genereeritud teed. Selleks kasutatakse hästi skaleeruvat (keskkonna mõõtmete suurenemise mõttes) algoritmi, mis produtseerib perekonna suboptimaalseid teid. Kõik teed on üksteisest võimalikult erinevad ning katavad ära terve keskkonna. Selliste teede kasutamine tagab sihikindla keskkonna avastamise ning võimalikult ohutu tee leidmise.

Töö käigus on mini-robotiga Khepera läbi viidud üle 600 testjuhtumi erinevates (täiesti tundmatu, osaliselt teada, dünaamiline, staatiline) keskkondades. Uut suboptimaalsete teede genereerimisalgoritmi on testitud lühimate teede strateegia vastu. Katsed näitasid, et suboptimaalsed teed annavad vähemasti sama hea tulemuse, kui lühimad teed täiesti või osaliselt tundmatutes keskkondades. Välja on pakutud uus teede planeerimismeetod, mida kirjeldatud keerukas keskkonnas kasutades on robot võimeline adapteeruma, vähendama kollisiooni riski ning missiooni täitmiseks kuluvat aega.

Lisaks käsitletakse tundmatu ja robotile ohtliku keskkonna uurimise kasulikkust missioonidel, mille kestvus on ajaliselt piiratud ning pakutakse selleks uut võimalikku heuristilist teede planeerimise lahendust.

ACKNOWLEDGEMENTS

I wish to express my highest appreciation and gratitude to my supervisors Maarja Kruusmaa and Jan Villemson for the encouragement, suggestions and all-around support necessary for this work.

I also thank my colleague Alvo Aabloo, who has made a lot of organisational work and has been a valuable buffer between different departments of University of Tartu and the third party.

I would also like to thank my chiefs Monika Oit and Uuno Puus at Cybernetica AS, who supported my studies in parallel to the commercial software development.

Also, I would acknowledge my financial support by Estonian Science Foundation grant No. 5613 and Tiger Leap Program of The Estonian Information Technology Foundation.

And last but not least – I would like to thank all my family members and friends for their tolerance while seeing me as often as an eclipse of the moon during the past months. When this thesis will be ready, I promise to have more time for my wife Eve and my baby son Iko.

APPENDIX A

M.Kruusmaa, J.Willemson, K.Heero. Path Selection for Mobile Robots in Dynamic Environments. In *Proceedings of the 1st European Conference on Mobile Robots (ECMR'03)*, pages 113-118, Radziejowice, Poland, September 2003.

Path Selection for Mobile Robots in Dynamic Environments*

Maarja Kruusmaa
Maarja.Kruusmaa@ide.hh.se
Halmstad University,
School of Computer Science,
Electrical and Electronic Engineering
BOX 823, Halmstad, Sweden

Jan Willemson
jan@cs.ut.ee
Tartu University,
Dpt. of Computer Science
Liivi 2, Tartu, Estonia

Kristo Heero
kristo@math.ut.ee
Tartu University,
Dpt. of Computer Science
Liivi 2, Tartu, Estonia

Abstract. This paper evaluates a path selection algorithm for mobile robots in large dynamic environments. The aim of the work is to reduce the risk of collisions and time of path following in cases when the robot repeatedly traverses between predefined target points (e.g. for transportation or inspection tasks). The algorithm is usable even if very little is known about the environment or if it gets completely restructured during the mission. This article concentrates on evaluating the performance of the algorithm. The tests are completed on the mini-robot Khepera that operates in an unknown dynamically changing environment. The test results show that the path selection algorithm is able to reduce collision risk, travel time and travel distances as well as increase the predictability of robot's behaviour and its degree of autonomy.

1 Introduction

The approach presented in this paper is motivated by the fact that many mobile robot applications imply repeated traversal between predefined target points in a dynamically changing environment. Examples of this kind of implementations are fetch-and-carry task of industrial and agricultural applications or visiting certain checkpoints in security and surveillance applications.

An efficiently operating robot is expected to fulfill its assignment as fast and as safely as possible. It means that it is worthwhile to avoid situations where the robot is forced to replan its route, take a detour, can drive into a deadlock or collide with unexpected obstacles.

Real world environments are dynamic by nature. Therefore, all the possible situations that can delay the robot or imply a hazard cannot be foreseen. However, by modeling the environment or learning its properties, the time delays can be minimized and the risk can be reduced.

1.1 Problem Statement

It is further assumed that:

1. The environment is dynamic and large. It is not possible or feasible to model it precisely or keep the model constantly updated.
2. The environment contains obstacles with unknown size and location. Traversing this environment implies risk of colliding with these obstacles, being delayed when maneuvering around them or ending up in a deadlock.
3. Sensorial capabilities of the robot are insufficient to distinguish between static, dynamic and semi-dynamic obstacles (e.g. between pillars and people, steady and replaced furniture).
4. The robot is working under time constraints and it has limited computational resources. It is expected to fulfill its mission as fast and safely as possible.
5. Localization errors are small and do not accumulate (like it is with GPS) and it is therefore possible to follow a preplanned path rather precisely.

* This research was supported by Estonian Science Foundation grant ETF5613.

The problem we aim at solving is the following: find paths between previously determined target points so that following them minimizes risk of collisions and speeds up the mission.

Our approach to the problem is based on the following observation. In a dynamic environment with an unknown obstacle distribution, the best path to the goal is not necessarily the shortest one. Depending on the nature of the environment, there may exist routes that are longer but easier to follow. By introducing a path generation algorithm, the robot can test several alternatives to reach the goal. By remembering its path following experiences, it can learn to follow paths that save time and reduce risk. As the environment changes, the robot will reevaluate its past experience and adapt to use new easily traversable paths.

1.2 Related Work

For a mobile robot operating in an uncertain environment, a natural requirement is obstacle avoidance (e.g. several implementations of potential fields [3]). However, obstacle avoidance, no matter how good it is, always implies a collision risk due to the uncertainty in sensor readings, motion planning, obstacle position or computational imprecision.

To minimize collision risk and time delays it is therefore important to select paths where as few obstacles as possible are encountered. Very few research studies reported so far consider the problem of path selection in dynamic environments [1, 2]. Unlike these approaches, we do not assume that the structure of the environment is known a priori.

The other difference from the above-cited approaches is that instead of the topological map we use a grid-based map to model the environment. The advantage of the grid-based map in this context is that the model does not have to be reorganized when the world changes and it therefore permits the robot to learn and adapt even in situations when very little is known about the environment or when the whole structure of the environment changes.

In our own previous work, we have used an heuristic algorithm to generate new innovative paths [4]. The tests have revealed that the robot is quickly able to adapt to the changes in the environment. As a result, the risk of collisions, time delays and traveled distances are minimized.

However, the tests also pointed to the shortcomings of the approach. During the tests it was observed that many paths that the heuristic algorithm generated were similar to each other. The robot spent much time waiting for a different solution to be found. It also appeared that sometimes the robot got trapped to local minima – some paths that would have been easy to follow where never generated.

To overcome this problem we created a new algorithm for paths generation [6]. The aim was to generate a minimal cover of the solution space. We proved that we can seed the memory with a relatively small number of alternative solutions and can keep it constrained without loosing the robots ability to learn and generalize.

This paper tests another, improved path selection mechanism. In the next section, we describe the algorithm and motivate our approach. Next, we describe some experiments with the mini-robot Khepera to compare our approach with the traditional shortest-path following and demonstrate the advantages of our approach. Finally, we end this paper with conclusions and directions for the future work.

2 Path Selection in Dynamic Environments

Figure 1 describes building blocks of our approach. The global planner receives tasks from the user. The tasks are requests to move to a specific point from its present location. Given a new task, the global planner can trust on its ealier experiences stored in the memory and use paths that have proven to be easy to traverse. If the properties of the paths in the memory are not good enough (e.g. too risky, long or crooked), the global planner can use the path generation unit to suggest new solutions for traversing to the given goal.

The task of the path generation unit is to generate innovative solutions to the path planning problem using the map of the environment. If a new solution turns out to be good, it can be stored in the memory and used later when the same problem will be encountered.

The chosen path will be presented to the low-level planning and execution unit that is responsible for task decomposition (if necessary), replanning, localisation, sensor data processing and actuator control.

In this paper we concentrate on finding a good path generation mechanism that generates possibly dissimilar paths from a given start to a given goal thus permitting the robot to try as many alternative approaches in its search for better paths as possible.

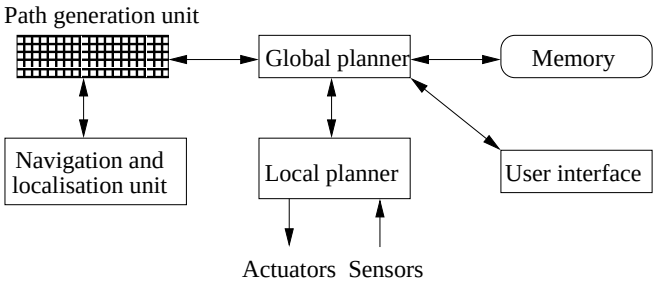


Fig. 1. General overview of path planning

Theoretically, the number of different paths on a grid-based map is overwhelming. There are too many alternatives to travel between two points and the robot could never try them all. In addition, most of those paths are infeasibly long, crooked and two difficult to follow. So the aim of the path selection algorithm is to:

1. generate paths that are easy to follow if they are free from obstacles;
2. generate paths that are as much different from each other as possible to let the robot find out as many innovative solutions as possible;
3. provide a mechanism that is able to discover virtually all possible alternatives;
4. cover the whole space of innovative solutions with as few alternatives as possible in order to maintain the robot's ability to generalize and keep the memory constrained.

We propose a method that works by dividing the grid into paths segments and then generating paths that cover all these segments. Full description of the method and its formal analysis is presented in [5].

Figure 2 illustrates one possible cover of the 3×4 grid.

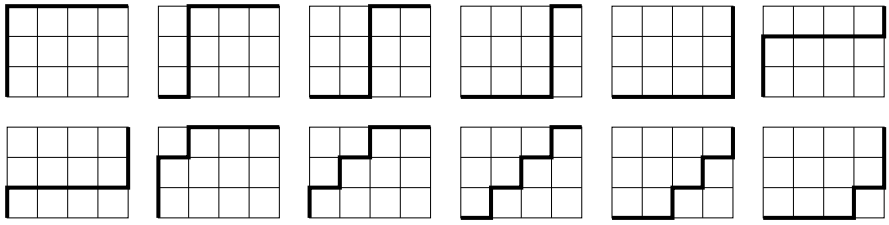


Fig. 2. Cover of the 3×4 grid

The paths selected by the robot are limited to those not having back turns and covering all the grid segments of length 2. In practice, paths relaxation is used to smoothen the paths and the zig-zags will be straightened.

It is proven in [5] that for a grid of the size $m \times n$, the cardinality of the minimal cover is $2m + 2n - 2$ paths. It means that the number of different paths is very small and grows linearly with a small constant, that makes it well scalable for very large domains.

While [5] gives a thorough insight to the theoretical aspects of the algorithm, this paper concentrates on demonstrating its advantages in practice. The following sections describe the test environment, experiments and compares the results of the path selection algorithm to the shortest paths approach.

3 Experimental Setup

The experiments are conducted using mini-robot Khepera. Khepera is a differencial drive miniature circular robot (with radius 26 mm) equipped with IR sensors for collision avoidance and it can be connected to a PC over serial link.

The size of our test environment is 1860×1390 mm. Since Khepera lacks sensors for accurate localization we solved this problem with a global vision system. The localization system is presented in figure 3. A video

camera is mounted to the ceiling to recognize the position of the robot. The PC processes the camera image to find robot's position and an algorithm running on it controls the robot over a serial link. In this way the localization errors are rather small (usually comparable to the size of the robot).

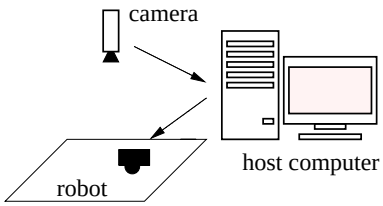


Fig. 3. Localization system

The environment is represented in figure 4, left. Figure 4, right, represents the same environment as shown from the overview camera.

Fig. 4. Environment in reality and as seen from the overview camera

The presence and location of the obstacles on the scene is randomly changed and unknown for the robot. The details of the obstacle distribution are presented in table 1. The frequency of obstacle replacement depends on its size, the 8 smallest obstacles are replaced after every traversal, and so the robot faces a changed environment for every traversal. An independent program using a random number generator determines the locations of the obstacles before every traversal.

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320 × 40	120 × 320 & 160 × 280	160 × 240 × 160	140 × 385	100 × 105	80 × 80	40 × 40
changing probability	0	0	0	0	0	0	1

Table 1. Obstacles

Figure 5 represents the model of the environment. The robot does not know anything about its environment except its size and the location of the target points. It has no idea about the presence and location of the obstacles, neither does it know about the program that determines the obstacle distribution. Therefore its model of the environment describes only the geometry of the environment and the location of its target points (marked with the letter G). Encountered obstacles that cause the robot to replan its route are marked with black rectangles. Since the robot cannot distinguish between frequently replaced and static obstacles, the obstacle map is cleaned after every traversal. The solid line represents the planned path from start to goal, the dotted line represents the actually followed path.

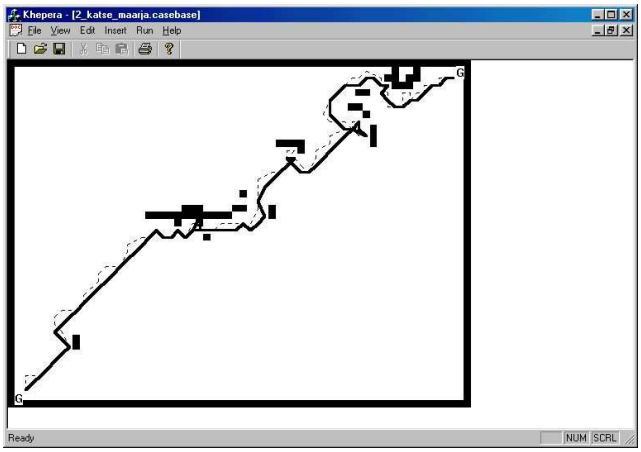


Fig. 5. Model of the environment

The mission of the robot consists of 50 tasks of traversing between the target points. Every task implies selecting a path, following the path, replanning on-line when obstacles occur and storing the parameters of the path when reaching the goal.

To evaluate the efficiency of our path selection algorithm, two missions consisting of 50 tasks are compared. The first mission is the conventional path following procedure. The robot always follows the shortest path to the goal. A wave transform algorithm is used to calculate the shortest path [7].

The second mission tests our path selection algorithm. When a new task is given, the robot either uses the path selection algorithm to generate an innovative path or selects an old path from its memory. The better the path in its memory is, the more likely it will be selected. The criterion of the path quality is the number of obstacles encountered on its way:

$$\text{path choosing probability} = \begin{cases} 1, & \text{if replannings} < 5; \\ 0, & \text{replannings} > 30; \\ 1 - \text{replannings}/30, & \text{otherwise.} \end{cases}$$

If the path selection algorithm is efficient, it helps the robot to find innovative paths that are easier traversable than the shortest path. As a result, the risk of collisions and time delays should be reduced.

4 Experimental Results

Table 2 represents our experimental results.

	replannings	time (s)	deviation (mm)	length (mm)	interruptions
shortest path algorithm	23,8	273,1	381,5	3878,5	20
path selection algorithm	9,9	191,2	281.0	3020,7	8
performance improvements	58%	30%	26%	22%	60%

Table 2. Experimental results

Five parameters were recorded and compared. The number of replannings is the parameter we explicitly wanted to minimize. Every time the robot detects an obstacle in its vicinity, it will replan its path. The number of replannings thus reveals the number of situations where a possible collision with an obstacle would have occurred. The test results show that the path selection algorithm reduced the number of possibly hazardous situation more than by half.

Although the paths generated by the path selection algorithm are longer than paths generated by the shortest path algorithm, they lead the robot faster to the goal. The time of path following was reduced

mainly because the robot did not spend so much time on replanning its route and maneuvering between obstacles.

The deviation from the originally planned path reveals the predictability of robot's behaviour. When the deviation is small, the robot followed more closely the route that it intended to. The path selection algorithm helped to reduce the deviation by 26%.

Although we did not aim at minimizing the traveled distance, on the contrary, longer paths were preferred if they were safer, the paths selection algorithm reduced the length of the paths by 22%. The reason is mainly that maneuvering around the obstacles increases the traveled distances.

During the tests, there were occasions when the robot got stuck in the corners of obstacles or did not manage to localize itself and therefore needed help from the experimenter. These cases were removed from statistics since the mission usually got interrupted and data was incomplete. However, it is important to notice that when the paths selection algorithm was used, the autonomy of the robot significantly increased. The occasions when the experimenter had to intervene were reduced by 60%.

Generally, it can be said that compared to the shortest path algorithm, the path selection algorithm improved all the measured parameters of the performance of the robot.

5 Conclusions and Future Work

This paper analyzed a path selection algorithm for repeated traversal in dynamic environments. The algorithm generates innovative paths between predefined target points and helps the robot to reduce time and risk of collisions. Compared to the other approaches to this problem, this approach is usable even when very little is known about the environment or when the environment is completely restructured during the mission. The disadvantage of the approach is that if very little is known about the surrounding, the robot needs a global localization system to keep track on its position.

The advantage of the current approach compared to our own earlier work is that we can prove theoretically the number of possible solutions and show that a relatively low number of solutions is needed to cover all path segments. An even more important theoretical outcome is that the algorithm scales up well to large domains. The number of solutions increases linearly with a small constant when the size of the environment increases. The practical advantage of this algorithm was supported by experiments. The algorithm helped to reduce the number of replannings, travel time, and distances as well as to increase the predictability of robot's performance and its degree of autonomy.

A possible direction of the future work could be to modify this approach to solve combined problems where the robot simultaneously works on two missions. The possible examples are carrying a load and cleaning the environment, or visiting checkpoints and inspecting the area at the same time. In these kind of missions it is favorable to use as different paths between target points as possible. The path selection algorithm will very efficiently cover the whole space even if the environment is large.

References

1. Karen Zita Haigh and Maria Manuela Veloso. Planning, execution and learning in a robotic agent. In *AIPS-98*, pages 120 – 127, June 1998.
2. H. Hu and M. Brady. Dynamic Global Path Planning with Uncertainty for Mobile Robots in Manufacturing. *IEEE Transactions on Robotics and Automation*, 13(5):760–767, October 1997.
3. J.S.Zelek. Dynamic issues for mobile robot real-time discovery and path planning. In *Proc. of Computational Intelligence in Robotics and Automation (CIRA'99)*, pages 232–237, 1999.
4. Maarja Kruusmaa. Global navigation in dynamic environments using case-based reasoning. *Autonomous Robots*, 14:71–91, 2003.
5. Maarja Kruusmaa and Jan Willemson. Algorithmic Generation of Path Fragment Covers for Mobile Robot Path Planning. Technical Raport, 2003.
6. Maarja Kruusmaa and Jan Willemson. Covering the Path Space: A Casebase Analysis for Mobile Robot Path Planning. *Knowledge-Based Systems*, 2003. Elsevier Science, to appear.
7. A. Zelinsky. Using path transforms to guide the search for findpath in 2D. *The Int. Journal of Robotics Research*, 13(4):315–325, August 1994.

This article was processed using
the TEX macro package with ECMR2003 style

APPENDIX B

K.Heero, J.Willemson, A.Aabloo, M.Kruusmaa. Robots Find a Better Way: A Learning Method for Mobile Robot Navigation in Partially Unknown Environments. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 559-566, Amsterdam, The Netherlands, March 2004.

Robots Find a Better Way: A Learning Method for Mobile Robot Navigation in Partially Unknown Environments*

Kristo HEERO[1], Jan WILLEMSON[1], Alvo AABLOO[2] and Maarja KRUUSMAA[2†]
[1]*Dept. of Computer Science, Tartu University, Liivi 2, 50409 Tartu, Estonia*
[2]*Institute of Technology, Tartu University, Vanemuise 21, 51014 Tartu, Estonia*

Abstract. This paper represents a method for mobile robot navigation in environments where obstacles are partially unknown. The method uses a path selection mechanism that creates innovative paths through the unknown environment and learns to use routes that are more reliable. This approach is implemented on Khepera robot and verified against shortest path following by wave transform algorithms. Based on the experimental data, we claim that robot's trajectory planned by wave transform algorithms is difficult to predict and control unless the environment is completely modelled and the localisation errors are small. We show that even small unmodelled obstacles can cause large deviation from the preplanned path. Our complementary approach of path selection decreases the risk of path following and increases the predictability of robot's behaviour.

Introduction

Mobile robots in human inhabited environments are expected to navigate safely and reliably as well as minimize travel time and energy consumption. Since real-world environments are complex, often unstructured and dynamic, it is impossible to build a complete model of robot's surrounding and keep it up to date. The robot is thus expected to operate as efficiently as possible with a rather limited amount of information.

Until now, research in mobile robot path planning has focused on finding optimal routes from start to goal. The optimality is usually measured in terms of travelled distances [1]. Other measures are also used, e.g. confidence value [2]. For planetary rovers the efficiency of a path is often expressed in terms of slope or roughness of the surface [3, 4].

Robots use local replanning to avoid unexpected obstacles in partially unknown environments. Since local planners do not use global knowledge, the behaviour of the robot is not globally optimised. Salich and Moreno have referred to this problem as to the dilemma of authority vs. freedom [5]. The dilemma rises from the fact that classic planners produce rigid orders while the behaviour of local reactive planners is unpredictable. Some researchers try to overcome this problem by incorporating global information to local decision making [6, 7].

Path planning algorithms used in robotics have been proven to give a globally optimal solution in globally known static environments. Their efficiency is not investigated in

* This research is supported by Estonian Science Foundation grant ETF5613.
† Corresponding author. E-mail: maarja.kruusmaa@ut.ee

complex, dynamic and partially unknown environments during long periods of time. Our experimental data suggests, that the dilemma local vs. global decision making is not so important as it is anticipated e.g. in [8]. It rather appears that if the global planner does not have all the global information about the environment. It anyway fails to create globally optimal plans.

Based on our experimental data we conclude that the environment has a much more significant effect on the behaviour of the robot than the algorithm used. Even if the robot always replans globally and always uses all the global knowledge available, it has a minor effect on the total outcome unless the environment is completely modelled. Our tests also show that robot's trajectory is difficult to predict and control. Even small unmodelled obstacles can considerably deviate the robot away from its globally planned path.

A good characteristic of a learning system is the predictability of its behaviour. The systems that better predict the outcome have learned the environment better. A mobile robot can predict its behaviour when it knows its position with a great certainty after a certain period of time. The ability to predict the trajectory makes it possible to optimise other parameters like travel time or energy consumption.

The problem we try to solve is thus how to optimise the behaviour of the robot in a partially unknown environment during a long period of time. There are two complementary approaches to increase the predictability of robot's behaviour. It is possible to gather more information about the environment to plan optimal paths. But since our experiments show that even small imprecision in input data or noise can considerably affect the robots trajectory, we have chosen an opposite approach. Instead of trying to model the environment we look for trajectories in a partially unmodelled environment that can be followed with a great precision.

We propose a method of covering a rectangular grid-based map with suboptimal paths. Previously we have described the method in detail [9] and have proven that the number of possible trajectories grows linearly with a small constant when the size of the map is increased. Therefore the method we describe can be used even in large-scale environments. The robot will then try to follow these paths and memorise them until it finds a trajectory that is sufficiently stable and easy to follow.

In our previous work [10], we have tested or approach in a totally unknown changing environment. The results show that the robot is able to adapt to the changes when the unknown obstacles are frequently replaced and learns to use trajectories that take it safer to the goal.

In this paper we report a series of tests to investigate the robot's behaviour in partially known environments. The environment is static to show the cause-effect relationship between the model of the environment and the robots behaviour. It allows us to draw a conclusion that the behaviour is influenced by the environmental model and the path planning algorithm but not by the robot's ability (or inability) to adapt to the changes.

Our paths selection algorithm is verified against shortest path following by a wave transform algorithm of [11] with global replanning.

Our initial hypothesis was that the shortest path following with global replanning would soon outperform our method when the environment becomes better known and when the unknown obstacles are smaller. We guessed that the shortest path planner would find the optimal path more likely if it knows the environment better. Tests did not confirm that hypothesis. On the contrary, the experimental data shows that wave transform algorithms are very sensitive to small imprecision in an environmental model. Even small unknown obstacles (or possibly sensor noise) can cause large deviation from the originally planned path.

Our method of path selection has two limits. First, it assumes that the robot will repeatedly traverse between two entry points. This assumption makes it possible to try several alternative trajectories. Fortunately there are plenty of mobile robot applications (e.g. transportation, surveillance, convoying) that presume repeated traversal between specified target points.

Second, the robot needs a fairly precise positioning system to follow the trajectories it has planned. In our tests we use an overhead camera to determine the robot's pose. We therefore suggest that the method works equally well with a satellite or pseudolite-based navigation. Since we test our approach in an environment where some static obstacles are modelled, it is principally possible to use these objects as landmarks. Yet we do not have any experience on how the robot would behave when the localisation errors are large, like it often happens with landmark based navigation.

In the next section we form the problem and list the assumptions we have made. We then describe briefly our path selection mechanism. After that, we describe the experiments and draw conclusions based on the experimental data.

1. Problem statement

It is further assumed that:

1. The environment is dynamic and large. It is not possible or feasible to model it precisely and/or keep the model constantly updated.
2. The environment contains obstacles with unknown size and location. Traversing this environment implies risk of colliding with these obstacles, being delayed when manoeuvring around them or ending up in a deadlock.
3. Sensorial capabilities of the robot are insufficient to distinguish between static, dynamic and semi-dynamic obstacles (e.g. between pillars and people, steady and replaced furniture).
4. Mapping, path planning and localisation are not the main objectives of the robot. These are presumptions to make the successful completion of a mission possible. Therefore they cannot take all of time and the computational recourses. Some resources are also needed for the main task that should be fulfilled as fast and safely as possible.
5. Localisation errors are small and do not accumulate and therefore it is possible to follow a preplanned path rather precisely.

The assumptions 1 and 3 seem to contradict with the experimental design where the environment is actually kept static. However, a static environment is not the necessary precondition of the approach. The environment is kept static only to find out the causal relation between an environmental model and the behaviour of the robot.

The problem we aim at solving is the following: find reliable paths between previously determined target points so that following them minimises collision risk and speeds up the mission.

Our approach to the problem solving is based on the following observation: in a dynamic environment with an unknown obstacle distribution, the best path to the goal is not necessarily the shortest. Depending on the nature of the environment, there may exist routes that are longer but easier to follow in terms of time or safety. By introducing a path generation algorithm, the robot can test several alternatives to reach the goal. By remembering its path following experiences, it can learn to follow paths that save time and reduce risk. As the environment changes, the robot will re-evaluate its experience and will adapt to use new easily traversable paths.

2. Path selection

Theoretically the number of different paths on a grid-based map is overwhelming. There are too many alternatives to travel between two points and the robot could never try all of them. In addition, most of those paths are unfeasibly long, crooked and difficult to follow. So the aim of the path selection algorithm is to:

- generate paths that are easy to follow if free from obstacles;
- generate paths that are as much as possible different from each other to let the robot find out as many innovative solutions as possible;
- provide a mechanism that in practice is able to discover virtually all possible alternatives;
- cover whole space of innovative solutions with as few alternatives as possible in order to maintain the robot's ability to generalise and keep the memory constrained.

We propose a method that works by dividing the grid into paths segments and then generating paths that cover all these segments. The full description of the method and its formal analysis is presented in [8].

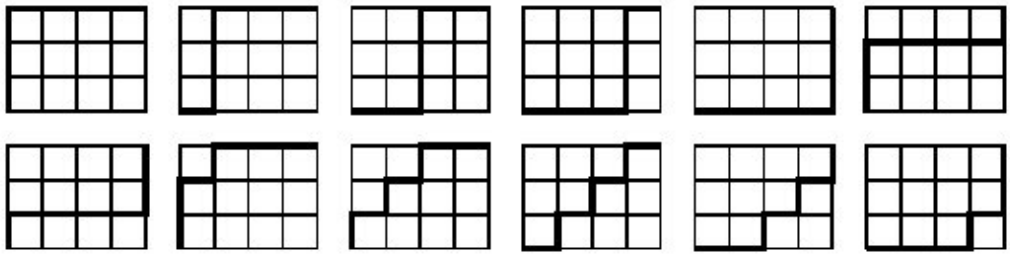


Figure 1: The cover of a 3×4 grid.

The paths selected by the robot are limited to those not having back turns and covering all the grid segments of length 2. Theoretically there are $2^{(m-1)(n-2)+(m-2)(n-1)}$ possible ways to cover a $m\times n$ grid with such a minimal cover. Figure 1 shows one possible cover of a 3×4 grid. In practice, paths relaxation is used to smoothen the paths and the zigzags will be straightened.

It is proven in [8] that for a grid of the size $m\times n$, the cardinality of the minimal cover is $2m+2n-2$ paths. It means that the number of different paths is very small and grows linearly with a small constant, which makes it well scalable for very large domains.

3. Experimental Design

The experiments are conducted using a mini-robot Khepera. It is a differential drive miniature circular robot (with radius 26 mm) equipped with IR sensors for collision avoidance and it can be connected to a PC over a serial link.

The localisation system is presented in Figure 2. A video camera is mounted to the ceiling to recognise the position and orientation of the robot. The PC processes the camera image to find robot's position and a computer algorithm controls the robot over a serial link. In this way the localisation errors are rather small (usually comparable to the size of the robot).

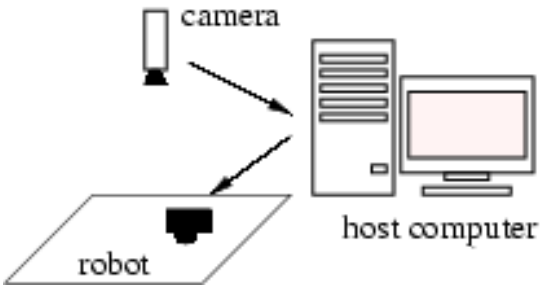


Figure 2. Localisation system.

The size of our test environment is 1860×1390 mm. It is represented in Figure 3 to the left. The picture in the middle represents the same environment as shown from the overview camera. The picture to the right in Figure 3 is the graphical interface of the computer program that controls the robot and monitors its behaviour.

Figure 3. The test environment (to the left), the same environment seen through the overview camera (in the middle) and as modelled by the control program (to the right).

The robot traverses repeatedly between the lower left corner and upper right corner of the environment in Figure 3. The physical environment for all test runs is the same but the environmental model varies. Figure 4 represents 3 different maps that are used to determine how much the environmental model affects the results.

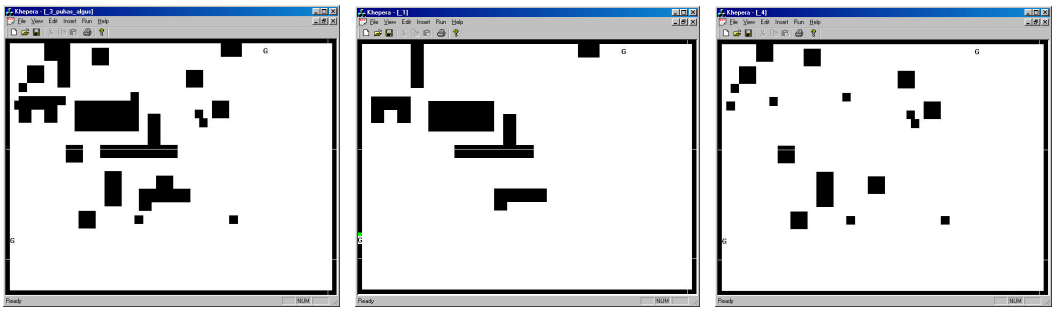


Figure 4. Environmental models used in experiments: a fully known environment (to the left), environment with large obstacles modelled (in the middle) and with small obstacles modelled (to the right).

The map to the left of Figure 4 is the precise model of the environment, containing the precise location of all obstacles. The map in the middle models only large obstacles while the location of small obstacles in unknown. The map to the right models only small obstacles while the large obstacles are unknown.

We compare our path selection method to shortest path following by a wave transform algorithm[10] with global replanning. Table 1 shows the number of trials with every environmental model with both path planning algorithms, shortest path planning vs. path selection. The number of trials depends on how fast the process stabilises.

Table 1. Number of trials.

Environmental model	Nr. of trials	
	Path selection	Shortest path
1.All obstacles known		10
2.Large obstacles known	20	50
3.Small obstacles known	20	50

The efficiency of the path planning algorithm is characterised by four parameters: number of replannings, travel time, travel distance and deviation from the originally pre-planned path.

One trial means planning a path from the lower left corner of the test environment to the upper right corner (or back again), following this path, replanning when an unknown obstacle is detected and recording the data when the robot reaches the goal.

The shortest path planning algorithm is the following:

1. Plan off-line a path from current start to current goal. This path is the shortest path to the goal calculated by a distance transform method [10].
2. Follow the path.
3. If an obstacle is detected plan a new path from its current position to the goal by a distance transform algorithm.
4. Repeat steps 2 and 3 until goal is reached.
5. Record travel time, travel distance, number of obstacles detected and deviation from the path planned at step 1.

The path selection algorithm is the following:

1. At the first trial select a suboptimal path planned by the method described in Section 3.
2. Follow the path.
3. If an obstacle is detected plan a new path from its current position to the goal by a distance transform algorithm.
4. Repeat steps 2 and 3 until goal is reached.
5. Smoothen the actually followed path to remove cycles, zigzags and gaps caused by localisation errors.
6. Store the smoothened path together with the travel time, distance, number of replannings and deviation.
7. At next trial check if there is a stored path with acceptably low number of replannings. If yes, follow this path. If no, choose a new path by using a method described in Section 3.
8. Repeat steps 2 to 7.

4. Experimental Results

All data from experiments, including recorded parameters at every trial, snapshots of every followed path and code of the control program are available at *http://math.ut.ee/~kristo/khepera/*. We here represent only some general statistics to compare the path planning strategies described above.

Table 2 represents data on the shortest path planning experiment. Table 3 represents data on path planning with path selection.

The efficiency of the path selection mechanism in case of a small number of trials largely depends on how fast the robot finds a suboptimal path that is easy to follow. While

running the test in the 3rd environment (with small obstacles known) the robot found an easy-to-follow suboptimal path at the first trial. For the sake of an unbiased interpretation we also represent data of another experiment that shows the worst case we have encountered. The robot had to try 4 suboptimal paths before it found one that was good enough. The last row of Table 3 therefore gives two figures for every parameter, the best result vs. the worst result.

Table 2. Results of shortest path planning.

Environmental model	Nr. of replannings	Travel time	Travel distance	Deviation from the preplanned path
1.All obstacles known	0.3	104	2555.0	43.8
2.Large obstacles known	12.7	123.3	2697.3	114.7
3.Small obstacles known	14.8	134.3	2768.0	107.0

Table 3. Results of planning with path selection.

Environmental model	Nr. of replannings	Travel time	Travel distance	Deviation from the preplanned path
1.All obstacles known				
2.Large obstacles known	0	104.1	2584.6	29.2
3.Small obstacles known	0/5.7	129.8/123.5	2534.2/2805.5	29.2/145.0

5. Discussion and conclusions

The first trials test the shortest path following strategy in a completely known environment (the first row in Table 2). It is the ideal case where globally best paths are planned with all available information. A closer look to the statistical data (available at the website) shows that the behaviour of the robot is predictable and stable. It means that we are able to control the robot with the great precision. Localisation errors, imprecision of mechanical linkages and sensor noise have no significant effect to the test results. Keeping all other things equal and changing only the environmental model or the path planning algorithm we can claim that the changes in experimental results are caused by one of the latter reasons.

Next we have verified the behaviour of the robot using two path planning strategies. Speaking in terms of decision-making theory, in case of shortest path planning, the robot can be described as a rational utility maximising agent. It always tries to find the shortest path to the goal considering all information available. In the case of path selection, the robot can be described as an explorative agent. It randomly tries suboptimal solutions to escape the local minimum and find a globally best solution.

The results show that by and all, the explorative agent is more successful. The advantage is apparent despite that the number of trials with the path selection method is smaller than the number of trials with the shortest path algorithm. Since the environment is static, larger number of trials would simply increase the advantages of the path selection mechanism since the robot would use the already found good solutions. At the same time the robot using the shortest path planning strategy does not learn and its behaviour never stabilises.

Another conclusion is that as soon as the environment is not modelled completely, the trajectory of the robot is hard to predict and control. Table 2 shows that small obstacles can

cause large deviation than large ones. The path selection algorithm represented here is one possibility to find reliable trajectories that increase the predictability of robot's behaviour.

Finally, we conclude that shortest path planning is not a relevant problem in partially unknown environments. As soon as the robot does not have all global knowledge available, suboptimal solutions give at least as good results as the optimal one. In order to increase the reliability of mobile robot applications, much more importance should be paid on modelling the environment and its changes.

This study obviously raises a question of how well the test environment models a real large-scale dynamic environment and how much Khepera can be considered as a model of a real robot. We suggest that the first question can be answered positively since this study focuses rather on modelling than navigation. We therefore expect the main conclusion that the environmental model plays a more important role than the path planning strategy, to hold also in real-world applications. However, we cannot be certain how much this conclusion can be extended to nonholonomic robots that due to their kinematics are not able to follow any possible pre-planned trajectory.

References

[1] A.Yahja, S.Singh, A.Stentz, "An Efficient on-line Path Planner for Outdoor Mobile Robots". *Robotics and Autonomous Systems*, 32, pp. 129-143, Elsevier Science, 2000.

[2] U.Nehmzow, C.Owen, "Robot Navigation in the Real World: Experiments with Manchester's Forty Two in Unmodified Large Environments", *Robotics and Autonomous Systems,* 33, pp.223-242, Elsevier Science, 2000

[3] A. Howard, H.Seraji, "Vision-Based Terrain Characterization and Traversability Assessment", *Journal of Robotic Systems*, Vol. 18, No.10, pp. 577-587, Wiley periodicals, 2001.

[4] D.B.Gennery, "Traversability Analysis and Path Planning for Planetary Rovers", *Autonomous Robots*, Vol. 6. pp. 131-146, Kluwer, Academic Publishers, 1999..

[5] M.A. Salichs and L. Moreno. "Navigation of Mobile Robots: Open Questions", *Robotica,* Vol. 18, pp. 227-234, Cambridge University Press, 2000.

[6] H.Seraji, "New Traversability Indices and Traversability Grid for Integrated Sensor/Map-Based Navigation", *Journal of Robotic Systems*, Vol. 20, No.3, pp. 121-134, Wiley periodicals, 2003.

[7] A. Sgorbissa, R.Zaccaria, "Roaming Stripes: integrating path planning and reactive navigation in a partially known environment", *11th International Conference on Advanced Robotics - ICAR 2003,* Coimbra, Portugal, July 2003

[8] Robin R. Murphy, Ken Hughes, Alisa Marzilli and Eva Noll, "Integrating explicit path planning with reactive control of mobile robots using Trulla", *Robotics and Autonomous Systems,* Vol. 27, Issue 4, pp. 225-245, Elsevier Science, 1997.

[9] M.Kruusmaa, J.Willemson. "Algorithmic Generation of path Fragment Covers for Mobile Robot Path Planning", *Technical Report*, 2003.

[10] M. Kruusmaa, J.Willemson, K.Heero. "Path Selection for Mobile Robots in Dynamic Environments", *Proc. of the 1st European Conference on Mobile Robots,* pp. 113 - 118., Poland 2003.

[11] A. Zelinsky. "Using Path Transforms to Guide the Search for Findpath in 2D. *The Int. Journal of Robotics Research,* Vol. 3 No. 4, pp. 315-325, August, 1994.

APPENDIX C

K.Heero, A.Aabloo, M.Kruusmaa. Learning Innovative Routes for Mobile Robots in Dynamic Partially Unknown Environments. *International Journal of Advanced Robotic Systems*, 2(3):209-222, 2005.

Learning Innovative Routes for Mobile Robots in Dynamic Partially Unknown Environments

Kristo Heero, Alvo Aabloo, Maarja Kruusmaa
Institute of Technology, Tartu University
Vanemuise 21, 51014 Tartu, Estonia
maarja.kruusmaa@ut.ee

Abstract: *This paper examines path planning strategies in partially unknown dynamic environments and introduces an approach to learning innovative routes. The approach is verified against shortest path planning with a distance transform algorithm, local and global replanning and suboptimal route following in unknown, partially unknown, static and dynamic environments. We show that the learned routes are more reliable and when traversed repeatedly the robot's behaviour becomes more predictable. The test results also suggest that the robot's behaviour depends on knowledge about the environment but not about the path planning strategy used.*

Keywords*: path planning, mobile robots, dynamic environment, robot learning.*

1 Introduction

Mobile robots in human inhabited environments should operate safely and reliably. At the same time they are expected to minimise energy consumption, travel time and distance. Optimisation of these parameters implies that the robot must be able to predict its behaviour in a partially unknown and changing environment.

Robots use path planning algorithms to plan a path from start to goal. In dynamic environments the environment can change during path following. To avoid collisions with unknown obstacles robots use local replanning. While classic AI planners are used to produce the global path to the goal, local replanners usually act reactively.

Mobile robot path planning is considered to be a well-established field incorporating several efficient path planning techniques and their modifications [1, 2, 3, 4, 5]. These path planners are proven to give a globally optimal plan in a static completely modelled environment. Several implementations also prove that by combining global and local path planning methods mobile robots are

able to operate in dynamic environments [6, 7, 8, 9]. However, there is no study that investigates their efficiency during long periods of time and the predictability of their behaviour.

Some studies investigate the long-term behaviour of localisation methods [10], fault tolerance [11] and reactive behaviour [12] but efficiency of path planners is usually evaluated only by examining few trials. Results prove that a robot using path planning methods is able to negotiate unknown obstacles and find a way to the goal but do not address the problem of the stability of its behaviour and optimality of solutions.

The reason for it might be that most of mobile robot path planners originally stem from the research with robot manipulators. The algorithms worked perfectly in small-size well-defined worlds and were easily adapted for 2D environments. As the field of mobile robotics developed the test environments grew large and more complex. Local replanning techniques and probability maps were introduced to cope with the new situation. At the same time global behaviour of path planning algorithms have remained unattended.

Predictability of robot's behaviour is an important parameter of a mobile robot. A robot that can predict its behaviour can estimate its position after a certain period of time. Increased predictability, in turn, makes it possible to optimize other parameters like travel time, distance, energy consumption, collision risk, etc. For robots working in a team predictability of each other's actions is inevitable. A robot interacting with a human is also expected to be stable and predictable. Knowing robot's location with a great precision makes it easier to find a lost robot in very large and complex environments.

The goal of this study is to investigate robot's behaviour in large and partially unknown environments as well as to find a way to increase predictability of robot's behaviour. We have set up a model environment for a mini-robot Khepera [13] and completed 600 test runs to investigate its long-term behaviour. We plan paths with wave transform algorithms widely used in mobile robotics [14]. The questions that we aim at answering are the following:

- How much does the efficiency of path planning algorithms depend on the accuracy of the world model?
- How much does explicit global replanning help to improve the performance of a path planner?
- How important is it to find a globally optimal path?
- Is it possible to increase predictability of robot's long-term behaviour if the environment is dynamic and partially unmodelled?

To answer the last question we introduce a learning method for fetch-and-carry tasks that looks for reliable trajectories in a partially modelled dynamic environment and compare its efficiency against shortest path following with a wavetransform algorithm. The results show that in a complex environment there may exist paths that are easier to follow than the shortest path. Finding these

innovative traits and following them helps to reduce collision risk as well as to minimize travel time, distance and deviation from the originally planned path.

To examine the performance of path planners we verify:

- Optimal path planning to suboptimal path planning
- Local path planning to global path planning
- Path planning in static and dynamic environments
- Path planning in known partially known and unknown environments.

The next section we describe finding innovative paths. The goal of this method is to find reliable trajectories to complete fetch-and carry tasks. Next, we describe the experimental setup. In section 4 we represent the experimental results. Section 5 discusses the experimental results.

2 Finding Innovative Routes

To illustrate the problem and to explain its relevance let us verify two test runs with the robot Khepera (Figure 1 and Figure 2). The robot has to traverse from the start point in the lower left corner of the map to the goal point in the upper right corner in both cases. The environment is completely unknown. Black rectangles on the grid map represent obstacles that the robot has detected during path following. Every detected obstacle causes the robot to replan globally.

Replanning is done with a wavetransform algorithm. The solid line on the figures is the path that the robot has planned, the dotted line is the actual path detected with an overview camera. The only difference is that while in Figure 1 the robot always plans the shortest path; in Figure 2 the robot first takes a suboptimal path to the goal until it counts the first unknown obstacle. At this point both of the algorithms become identical.

Eventually it appears that the robot in Figure 1 has taken a much longer route to the goal and deviated considerably from its original course. The suboptimal path in Figure 2 appears to be much shorter and easier to follow. Since the robot counts only few obstacles, collision risk is less. Replanning does not cause large deviation. Even when the robot replans it eventually drifts back to its original course.

Many mobile robot applications assume repeated traversal between predefined target points. The most common is a transportation task (fetch and carry), but also surveillance, guiding, rescuing etc. may fall into this category. If the robot was able to find stable trajectories like in Figure 2 then following them would considerably speed up the mission, reduce risk to the robot or to the environment and reduce energy consumption.

The problem is thus how to find trajectories that increase the predictability of robot's behaviour and reduce risk.

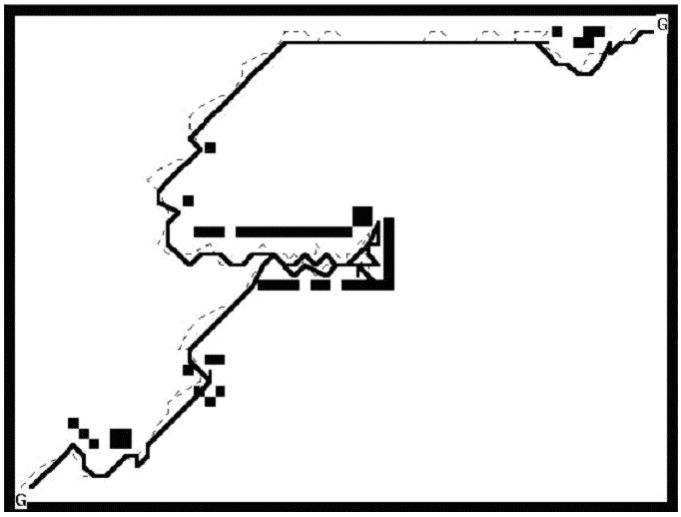


Figure 1. Path planning in an unknown environment. Shortest path planning with a wavetransform algorithm

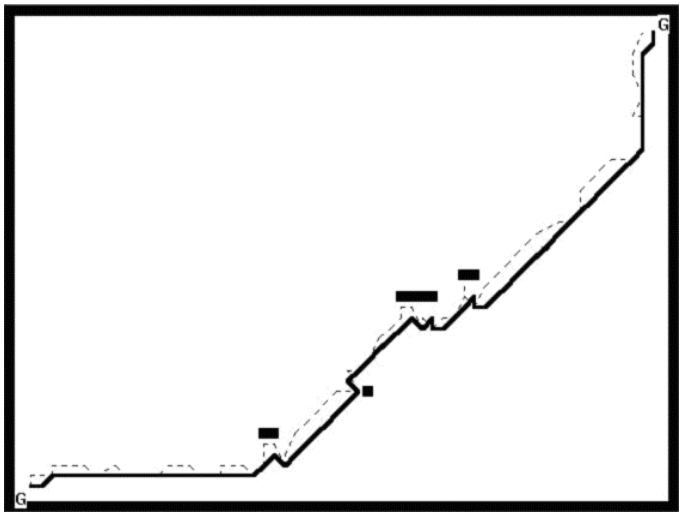


Figure 2. Path planning in an unknown environemnt. Suboptimal path following

One possibility would be to include information about reliability on the map. For grid-based maps, there exist reliable methods for incorporating ambiguity caused by sensor readings to the map [15]. Land-rovers also consider roughness and slope of the terrain to find the most reliable path [16, 17]. The optimal path is thus not expressed in terms of distance, but in terms of a cost function considering the traversability of the path. Cost functions are also used with topological path planning since exact distances between graph nodes are often unknown [18].

To increase reliability of path planning it would be necessary to include information about dynamic obstacles to the map. In [19] this is done by observing the obstacle distribution with an overview camera. A path planner then avoids areas that are more likely occupied. If it is not possible to observe the environment globally, the map is very difficult to keep up to date. Moreover, from Figure 1and Figure 2 it appears that some obstacles deviate the robot more away from its original path than others and to decrease deviation form the original path this information should also be taken into account.

To increase predictability of robot's behaviour, the grid map had to reflect the ambiguity caused by sensor readings, localization errors, dynamic obstacles and their effect to replanning procedures. To permit efficient path planning this map must be constantly updated. At the moment there is no such a method available that combines all these sources of uncertainty in a reliable manner and is easy to manage in real time.

To increase the predictability of path following for fetch and carry tasks we look at the problem from an opposite viewpoint. Instead of modelling the environment as accurately as possible we look for paths where inaccuracy of the world model does not significantly influence the result of path planning. Practically we choose a set of suboptimal paths and evaluate their traversability by trail-and error until we find some that satisfies our criteria of safety and reliability.

This approach is applicable for fetch and carries tasks since it implies repeated traversal between predefined targets. It also presumes that localisation errors are small and do not accumulate since the robot is expected to determine its position rather accurately. It is therefore easiest to apply the method if GPS or pseudolite navigation is used, for example like in [20].

2.1 Problem Statement

It is further assumed that:

1. The environment is dynamic and large. It is not possible or feasible to model it precisely and/or keep the model constantly updated.
2. The environment contains obstacles with unknown size and location. Traversing this environment implies risk of colliding with these obstacles, being delayed when manoeuvring around them or ending up in a deadlock.
3. Sensorial capabilities of the robot are insufficient to distinguish between static, dynamic and semi-dynamic obstacles (e.g. between pillars and people, steady and replaced furniture).
4. The environment is unstructured or the structure of the environment is unknown.
5. Mapping, path planning and localisation are not the main objectives of the robot. These are presumptions to make the successful completion of a mission possible. Therefore they cannot take all of time and the

computational recourses as some resources are also needed for the main task.

6. The robot is expected to fulfil its mission as fast and safely as possible.
7. Localisation errors are small and do not accumulate and therefore it is possible to follow a pre-planned path rather precisely.

The problem we aim at solving is the following: find reliable paths between previously determined target points so that following them minimises collision risk and speeds up the mission.

Our approach to the problem solving is based on the following observation. In a dynamic environment with an unknown obstacle distribution, the best path to the goal is not necessarily the shortest. Depending on the nature of the environment, there may exist routes that are longer but easier to follow in terms of time or safety. By generating innovative tracks with a path generation algorithm, the robot can test several alternatives to reach the goal. By remembering its path following experiences, it can learn to follow paths that save time and reduce risk. As the environment changes, the robot will re-evaluate its experience and will adapt to use new easily traversable paths.

2.2 Suboptimal Path Generation

Theoretically the number of different paths on a grid-based map is overwhelming. There are too many alternatives to travel between two points and the robot could never try them all. In addition, most of those paths are unfeasibly long, crooked and difficult to follow. So the aim of the path selection algorithm is to:

- generate innovative routes that are easy to follow if free from obstacles;
- generate paths that are as much as possible different from each other to let the robot find out as many innovative solutions as possible;
- provide a mechanism that in practice is able to discover virtually all possible alternatives;
- cover the whole space of innovative solutions with as few alternatives as possible in order to maintain the robot's ability to generalise and keep the memory constrained.

We propose a method that works by dividing the grid into paths segments and then generating paths that cover all these segments. The full description of the method and its formal analysis is presented in [21].

Figure 3 illustrates one possible cover of a 3×4 grid. The paths selected by the robot are limited to those not having back turns and covering all the grid segments of length 2. In practice, paths relaxation is used to smoothen the paths and the zig-zags will be straightened.

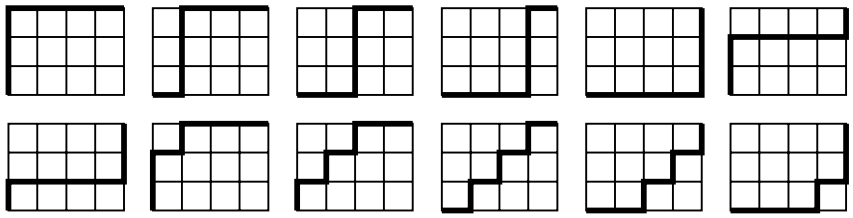


Figure 3. The cover of a 3×4 grid

It is proven in [21] that for a grid of the size $m \times n$, the cardinality of the minimal cover is $2m + 2n - 2$ paths. It means that the number of different paths is very small and grows linearly with a small constant, which makes it well scalable for very large domains.

2.3 Learning Innovative Routes

In the previous section we presented a method of covering the m × n grid map with a set of suboptimal path. We assume that although these paths are longer than the shortest one, in a dynamic and partially modelled environment some of these routes may be easier to follow. Since the map may be inadequate and does not contain information that permits to evaluate reliability of paths, we can evaluate their traversability only after following them. Therefore the application of the algorithm is also limited to missions that assume repeated traversal between predefined targets. The general form of the algorithm represented in Figure 4.

In the next section we describe the experimental design and after that represent the results of the experiments. The experiments were set up to verify the path selection algorithm against shortest path following, specifically to show how dynamic or how well modelled the environment can be for the path selection algorithm to outperform shortest path following. For the sake of completeness and in order to draw some more general conclusions we have also included some test runs published earlier in [22] and [23].

3 Experimental Design

3.1 Robot

The experiments are conducted using a mini-robot Khepera (see Figure 5). It is a differential drive miniature circular robot (with radius 26 mm) equipped with IR sensors for collision avoidance and it can be connected to a PC over a serial link.

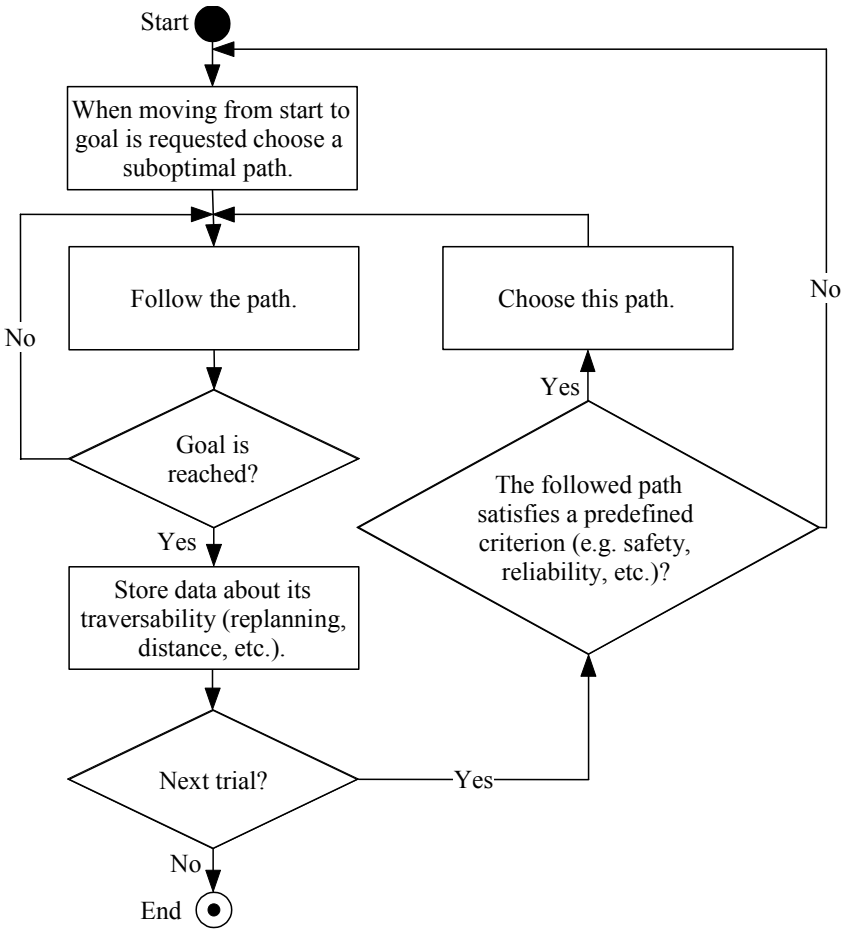


Figure 4. The path selection algorithm

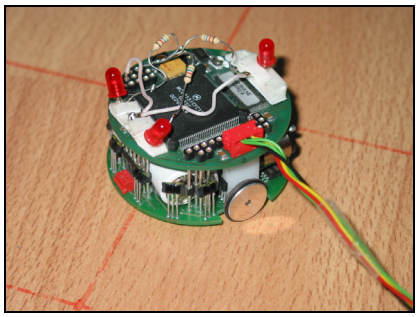


Figure 5. The Khepera robot. LEDs are mounted on its top to detect it with an overview camera

3.2 Localization

The localization system is presented in Figure 6. A video camera is mounted to the ceiling to recognize the position and orientation of the robot. The PC processes the camera image to find robot's position and a computer algorithm controls the robot over a serial link. In this way the localisation errors are rather small (usually comparable to the size of the robot).

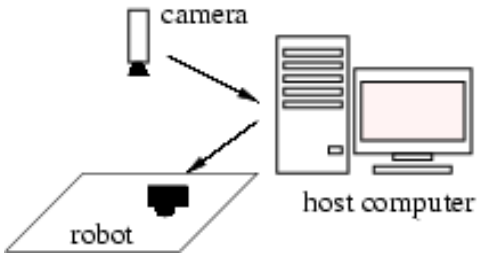


Figure 6. Localization system with an overview camera

3.3 Test Environments

To test the path planning algorithms we have set up a test environment represented in Figure 7. The size of the test environment is 1860 mm × 1390 mm and it contains replaceable obstacles. Figure 8 represents the same environment observed though the overview camera. Khepera is detected with the help of 3 LEDs mounted on its top. The robot is shown in the middle of the Figure 8, the serial cable connecting it to the computer is also visible. The isosceles triangle formed by the LEDs determines the position as well as orientation of the robot.

Figure 7. The test environment

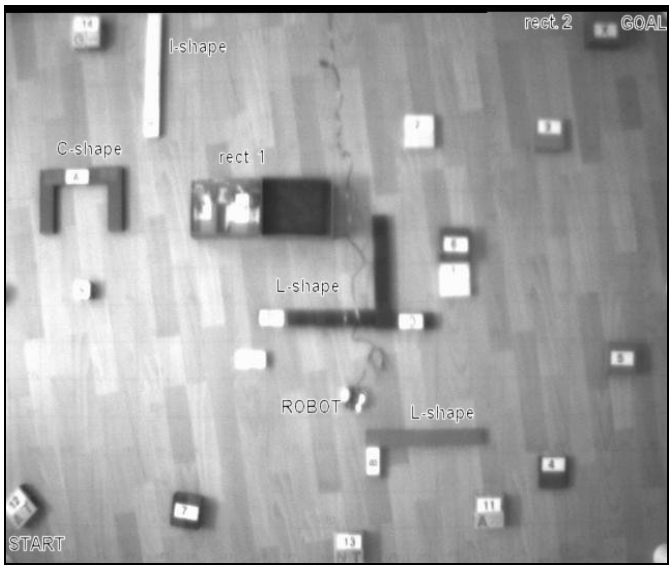


Figure 8. The test environment seen through the overview camera

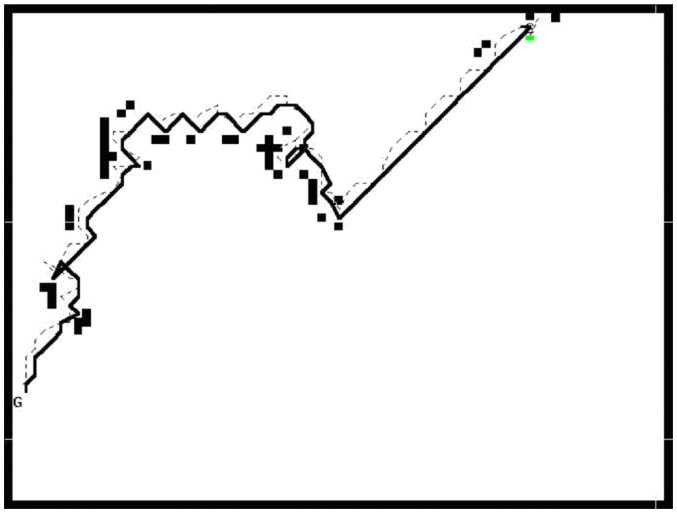


Figure 9. The user interface of the control program

A computer program controls the robot, detects its position with the overview camera and records its behaviour. The user interface of the program is shown in Figure 9.

The robot repeatedly traverses between the lower-left and upper-right corner of the test environment and back again. Black rectangles in Figure 9 are occupied cells; either modelled *a priori* or detected with the IR sensors of Khepera. Every obstacle that the robot detects forces it to replan its route

according to a previously determined strategy. The solid line is the planned route; the dotted line is the actual route of the robot recorded with help of the localization system.

3.4 Obstacles

The specification of the obstacles is given in Table 1. Their location is generated with a random number generator. The location of the larger obstacles specified as I-shape, L-shape, C-shape, *rect.1* and *rect2* is kept constant during all the test runs (Figure 8 shows their locations). In dynamic environments we change the location of obstacles *rect. 3* and *rect.4*. Keeping the location of the static obstacles constant during all the test runs makes it possible to verify the experimental results.

The performance of the robot is verified in 6 environments with a different degree of dynamicity and different amount of *a priori* known information. In next subsections, we describe in detail the path planning algorithms and test environments.

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40

Table 1. Specification of the obstacles

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40
changing probability	0	0	0	0	0	0	0
type	modelled	modelled	modelled	modelled	modelled	unknown	unknown

Table 2. Obstacles in environment 2

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40
changing probability	0	0	0	0	0	0	0
type	unknown	unknown	unknown	unknown	unknown	modelled	modelled

Table 3. Obstacles in environment 3

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320 ×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40
changing probability	0	0	0	0	0	0	0
type	unknown	unknown	unknown	unknown	unknown	unknown	unknown

Table 4. Obstacles in environment 4

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40
changing probability	0	0	0	0	0	0	1
type	unknown	unknown	unknown	unknown	unknown	unknown	unknown

Table 5. Obstacles in environment 5

	I-shapes	L-shapes	C-shapes	rect. 1	rect. 2	rect. 3	rect. 4
amount	2	2	1	1	1	10	8
size (mm)	320×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40
changing probability	0	0	0	0	0	0.2	1
type	unknown	unknown	unknown	unknown	unknown	unknown	unknown

Table 6. Obstacles in environment 6

3.4.1 Environment 1 – Static, Known

This environment is set up to test the performance of the robot in perfect conditions.

The environment 1 is the complete and correct model of the environment. Locations of all obstacles is precisely known and do not change. The model is represented in Figure 10. The start and goal points are indicated with a letter G.

3.4.2 Environment 2 – Static, Large Obstacles Known

The physical environments 1, 2 and 3 are identical, only the models of the environments differ. All the obstacles in this environment are kept static during the tests.

The environments 2 and 3 were set up to verify the performance of the path selection algorithm in a partially modelled world.

In environment 2 the large obstacles are known a priori while the presence and location of small obstacles is unknown (see Table 2 for details). The model of the environment 2 is shown in Figure 11.

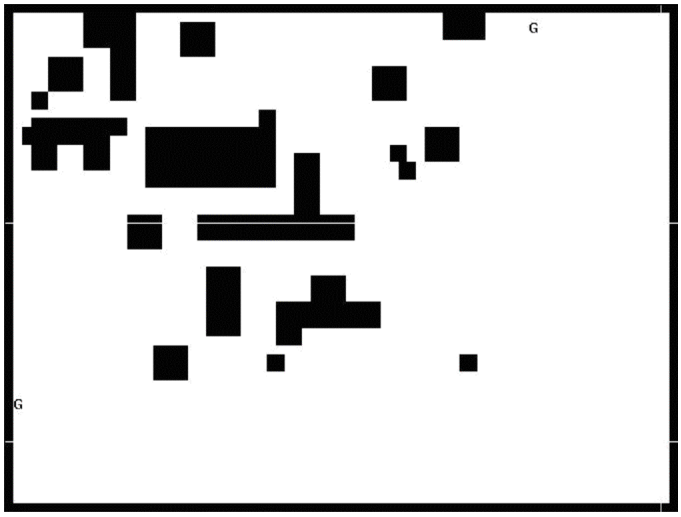


Figure 10. The complete model of the environment. The letter G indicates start and goal points

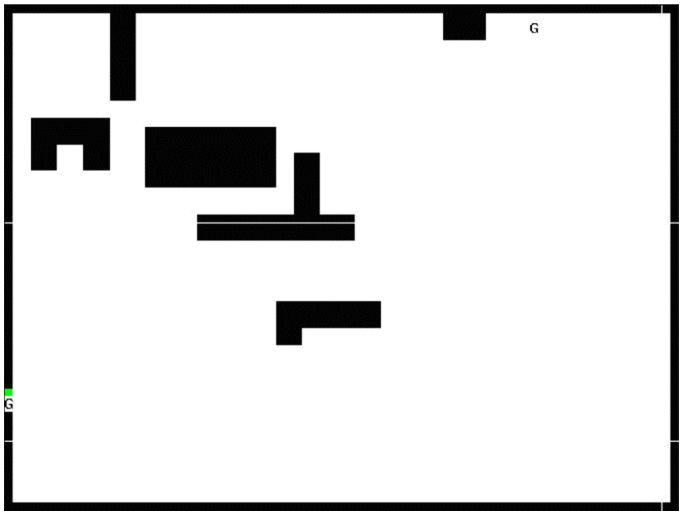


Figure 11. The partially modelled environment. Large obstacles are modelled

3.4.3 Environment 3 – Static, Small Obstacles Known

The model of the environment 3 is the reverse of the model of the environment 2. All obstacles that were known in environment 2 are unknown while all obstacles, which were unknown in environment 2, are known here (see Table 3). Like it was stated above, the physical environment that the robot traverses is the same for environments 1, 2 and 3. The model of the environment 3 is represented in Figure 12.

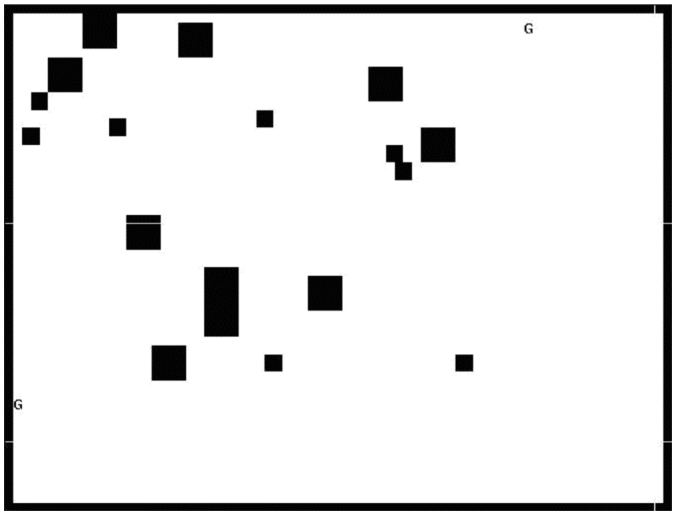


Figure 12. The model of the test environment with small obstacles modelled

Figure 13. Model of the unknown environment

3.4.4 Environment 4 – Static Unknown

The test environment 4 was set up to test the performance of the path selection algorithm in an extreme case where nothing is known about the environment except its size and goal locations.

The model of the environment is represented in Figure 13. All obstacles are static (see Table 4).

3.4.5 Environment 5 – Slightly Dynamic, Unknown

The environments 5 and 6 are dynamic and unknown. Tests in these environments evaluate how well does the path selection algorithm adapt to the environmental changes.

Environment 5 is unmodelled with the 8 obstacles replaced after every trial. The model of the environment is represented in Figure 13.

Obstacle distribution is presented in Table 5. Locations of all 8 obstacles specified as *rect. 4* are replaced before every trial with probability 1. Their new locations are generated randomly. If the new location of an obstacle coincides with an existing obstacle, a new location is generated.

A realistic environment can change during the path following. However, this is a feature that we were not able to model, since changing the environment during the path following would have disturbed the localisation system that uses an overview camera. We therefore changed the environment before every trial. In this way the robot faces a new changed world every time it travels to its goal location, but it does not observe the environment changing during the path following.

3.4.6 Environment 6 – Moderately Dynamic, Unknown

The model of the environment is the same as in Figure 11. Environment 6 is more dynamic as 18 obstacles are replaced after every trial. The obstacle distributions are represented in Table 6. Like in the environment 5, the obstacles specified as *rect. 4* are replaced with the probability 1.0. In addition 10 more, twice as large obstacles *rect. 3*, are replaced after every trial, each with a probability 0.2.

Dynamic obstacles actually correspond to those that are unknown in environment 2 while all static obstacles correspond to those that are known in environment 2.

3.5 Algorithms

This subsection describes path planning strategies used to evaluate the path selection algorithm. The performance of the path selection algorithm is verified against shortest path following with wavetransform algorithms. The wavetransform algorithm is selected as one of the commonly used path planning methods on a grid map. Also our own path selection algorithm uses the same wavetransform methods for replanning. Therefore we can assure that any differences in performance are caused only by the learning algorithm but not by the replanning method.

The robot uses 3 different path planning strategies:

1. Shortest path following always plans the shortest path to the goal from its current position.
2. Path selection with global replanning uses the path selection algorithm to look for suboptimal routes to the goal. It follows the suboptimal route

until the first unexpected obstacles is detected. Then the shortest path following is used to reach the goal.

3. Path selection with local replanning uses the path selection algorithm to find suboptimal routes to the goal. When unexpected obstacles are detected, local replanning is used to find its way back to the pre-planned route.

Next, we describe every path planning strategy in detail. The program code corresponding to every algorithm is available at *http://math.ut.ee/~kristo/khepera/*.

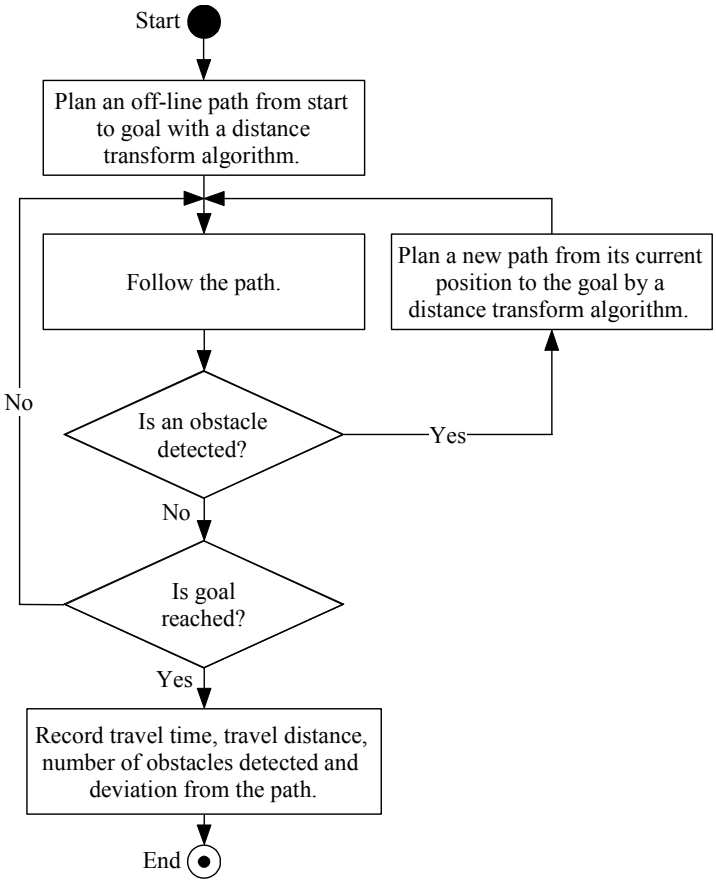


Figure 14. The algorithm of shortest path following

3.5.1 Shortest Path Following

Figure 14 describes the shortest path following strategy. A robot using this strategy always tries to find the shortest possible path to the goal considering all

global information available. To find the shortest path, the distance transform algorithm is used [14]. This algorithm always finds the globally optimal path to the goal and at the same time keeps the robot at the safe distance from obstacles. This strategy uses no memory to store the information about previously followed tracks.

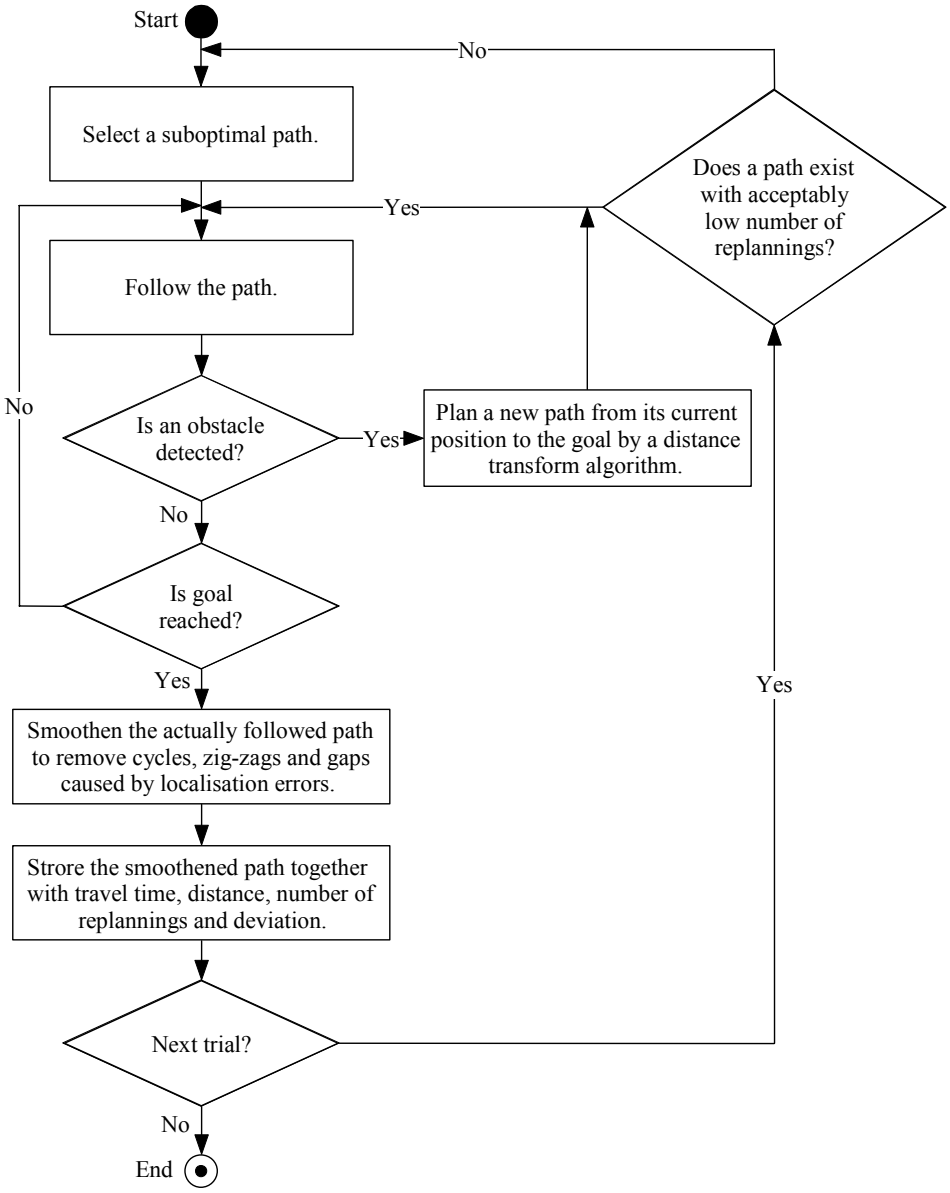


Figure 15. Path selection algorithm with global replanning

3.5.2 Path Selection With Global Replanning

Figure 15 represents the flow diagram of the path selection algorithm with global replanning. The bottom line of this path following strategy is to first travel a suboptimal route and if this route appears to be blocked with an obstacle, abandon the pre-planned route and turn to the shortest path following strategy.

The followed path is usually different from the pre-planned one because unknown obstacles deviate the robot from its initial course. It is also crooked since the robot repeatedly corrects its localization error and avoids collisions with obstacles. The zigzags of the followed path are therefore straightened and gaps and cycles are removed. This procedure is documented in [24]. When the goal is reached the followed path is remembered and stored together with statistical data (nr. of obstacles detected, travel time, distance, deviation from the original path).

This strategy minimizes the risk of path following; therefore it chooses routes that are most likely to be free from obstacles. The robot first looks for a stored path with an acceptably low number of replannings while repeatedly traversing between the target points. If such a path exists, it is repeatedly followed as long as it satisfies the robot's criterion of safety. Otherwise a new suboptimal route is chosen, followed and remembered. The criterion of path selection among the stored paths is the probability inversely proportional to the number of unexpected events:

$$probability = \begin{cases} 1, & replannings < 5 \\ 0, & replannings > 30 \\ 1 - replannings / 30, & \text{otherwise} \end{cases}$$

Equation 1.

Obviously, the efficiency of this algorithm depends on how fast the robot finds a route with acceptable parameters. When this route or several good routes are found, it usually keeps following them, robot's behaviour stabilizes and the overall performance increases. The final result thus largely depends on how many trials are performed and how lucky the robot is to find a good route. Longer test runs would thus increase the efficiency of the path selection algorithm compared to shortest path following strategy which does not learn and remember and therefore does not improve its performance. To avoid a biased interpretation of the test results we have therefore analyzed separately the parameters of innovative suboptimal routes and learned (repeatedly traversed) routes.

3.5.3 Path Selection With Local Replanning

This path planning strategy described in Figure 16 follows the initially chosen suboptimal route to the end. When the unexpected obstacle is detected, it plans

a path around it and tries to return to the pre-planned path. If new obstacles are detected during replanning, it always tries to turn back to the pre-planned path a little further away from the initial returning point. The local replanning algorithm is therefore somewhat smarter than the plain global task decomposition algorithm. It does not try to reach to a certain sub-goal that could be hard to achieve but chooses a new sub-goal. Put differently, it avoids sub-goal obsession, as R. Murphy calls it [25]. To replan locally the robot uses the same distance transform algorithm that is used for shortest path planning.

Like in case of the previous strategy, the test results depend on how lucky the robot is to find a good route and how long is the test run. Therefore in experimental results, the innovative suboptimal path and learned path are analyzed separately.

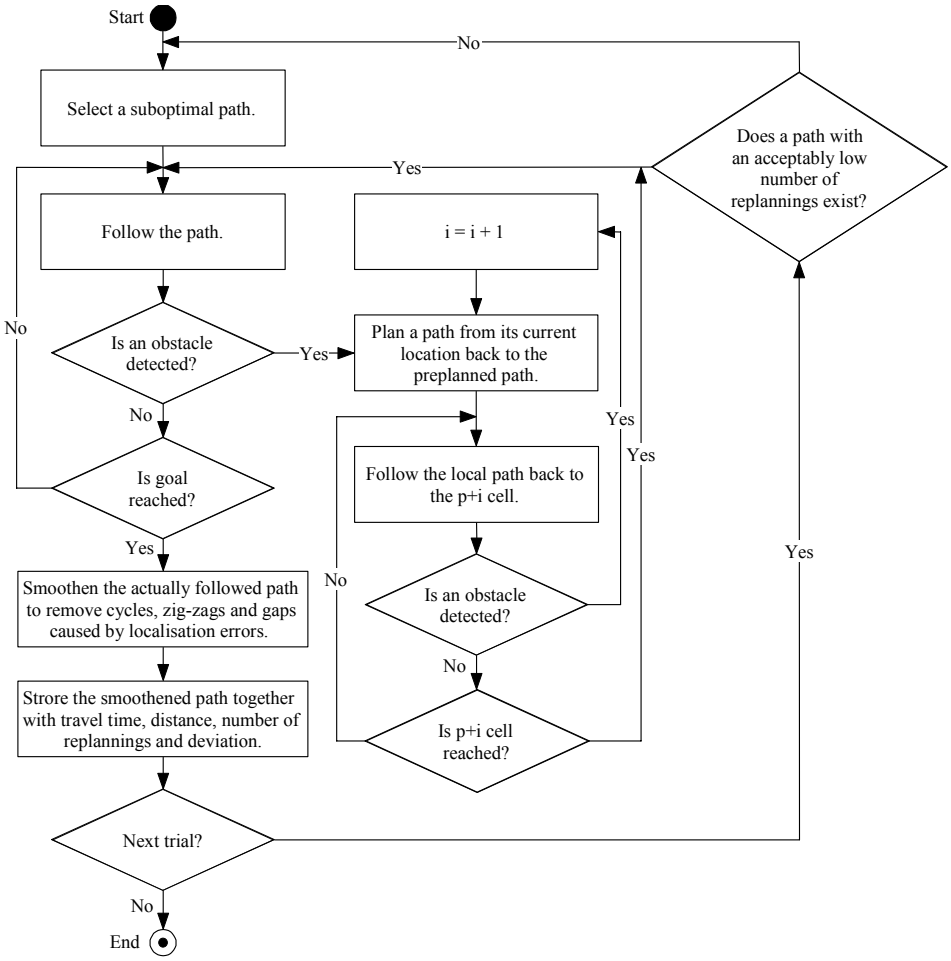


Figure 16. Path selection algorithm with local replanning

4 Experimental Results

4.1 Shortest Path Following in a Fully Modelled Static Environment

We here represent the detailed results of 10 test runs in a fully modelled static environment. This test evaluates the efficiency and stability of shortest path planning with a wavetransform algorithm and gives a reference for evaluating the performance in partially modelled and unknown environments.

One trial means planning a path from the start to the goal, following the path and replanning if necessary. In the end of every trail four parameters are recorded: number of replannings due to unexpected obstacles, travel time, deviation from the originally planned path and travel distance.

Table 7 represents detailed experimental results. Figure 17 represents the 5th trail recorded by the computer program and Figure 18 represents the 6th trial where the robot traverses in the opposite direction.

This data indicates that the program is able to control the robot rather precisely and the behaviour of the robot is predictable. The deviation from the original path is not more than by the diameter of the robot and only in one case a detected obstacle or sensor noise forces the robot to replan its course (trial nr. 3).

We can conclude that localisation errors, imprecision of mechanical linkages or the control program do no influence the test results significantly. Keeping all other things equal and changing only the model of the environment or the path planning algorithm we can thus claim that any changes in performance are caused by one of these reasons.

The tests represented in this paper consist of 600 trials. In the following subsections we represent only the average values of all the test runs. The detailed data about every trail as well as snapshots of every followed path similar to Figure 17 and Figure 18 are available online at *http://math.ut.ee/~kristo/khepera/*.

No. of trial	No. of replannings	Travel time (sec)	Deviation (mm)	Distance (mm)
1	0	105	29,2	2535,3
2	0	103	58,4	2551,2
3	3	108	29,2	2561,3
4	0	104	58,4	2576,9
5	0	102	29,2	2535,7
6	0	104	58,4	2592,7
7	0	104	29,2	2540,8
8	0	102	58,4	2540,8
9	0	105	29,2	2543,0
10	0	103	58,4	2573,0
Average	0,3	104	43,8	2555,1

Table 7. Shortest path following in a fully modelled static environment

Figure 17. Path followed in a fully modelled static environment. Robot moves from the lower-left corner to the upper-right corner

Environment		1	2	3	4	5	6
Shortest path following		10	50	50	50	50	50
Path selection with	Suboptimal path	-	1	5	16	14	31
global replanning	Learned path	-	9	25	34	36	19
Path selection with	Suboptimal path	-	-	-	8	28	17
local replanning	Learned path	-	-	-	42	22	33

Table 8. Number of trials

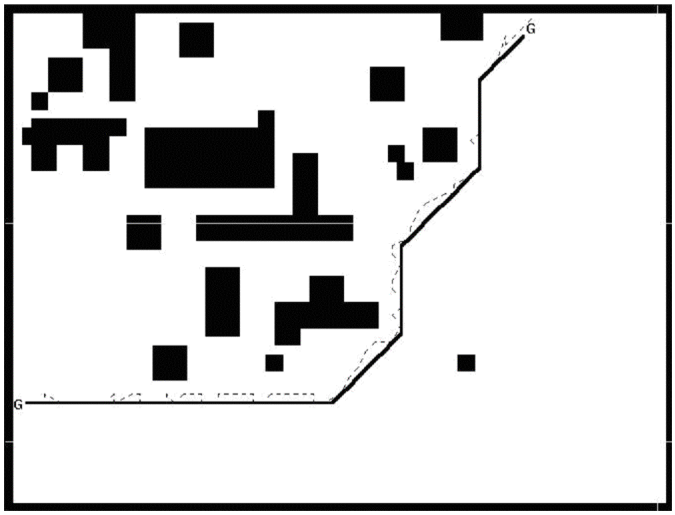


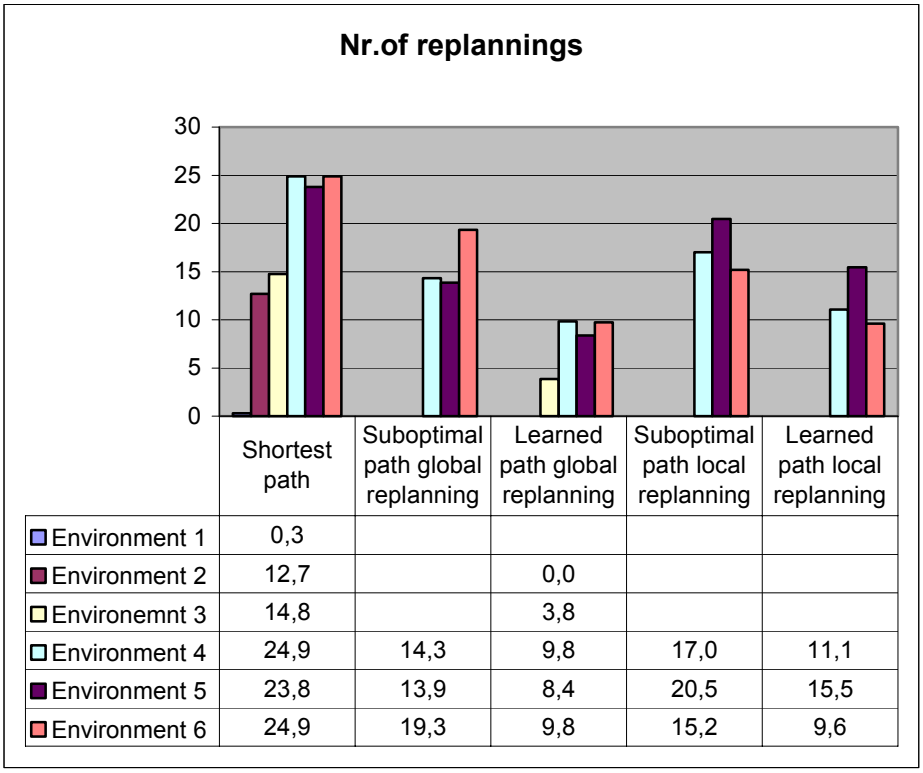
Figure 18. Path followed in a fully modelled static environment. Robot moves from the upper-right corner to the lower-left corner

The number of trials in every test environment is represented in Table 8. Except the trials in a static fully modelled environment where the process was very stable, all other tests to evaluate shortest path following consist of 50 trials.

Path planning with path selection algorithm in unknown environments also consists of 50 trials. As it was described above, the path selection algorithm tries several innovative suboptimal paths and learns to follow this suboptimal path, which are more stable. Learned suboptimal path and innovative suboptimal paths are therefore analyzed separately and depending on the environment the number of trials differ. However, the total number of trials is still 50 in unknown environments. Since in partially known environments the learning process stabilizes very quickly, the total number of trials is less (10 in environment 4 and 30 in environment 6).

4.2 Number of Replannings

Figure 19 represents the average number of replannings caused by unexpected obstacles. This is the parameter that the path selection algorithm explicitly minimizes and therefore learning considerably minimizes the risk of collisions. The number of replannings also decreases when the environment becomes better known, which is not a surprising result since the modelled obstacles can now be avoided in advance. The unexpected result is that following suboptimal path gives a better result than following the shortest one. Since the shortest path following algorithm that globally replans always maximizes the expected utility, one would expect that the total outcome of this strategy would be better.

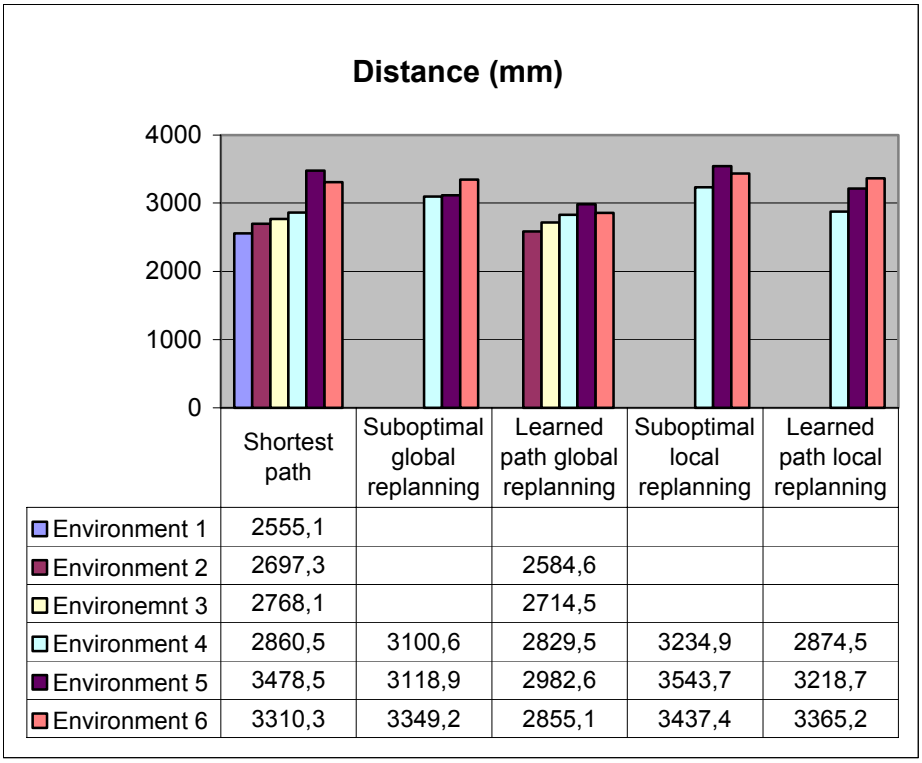


	Shortest path	Suboptimal path global replanning	Learned path global replanning	Suboptimal path local replanning	Learned path local replanning
Environment 1	0,3				
Environment 2	12,7		0,0		
Environemnt 3	14,8		3,8		
Environment 4	24,9	14,3	9,8	17,0	11,1
Environment 5	23,8	13,9	8,4	20,5	15,5
Environment 6	24,9	19,3	9,8	15,2	9,6

Figure 19. Average number of replannings caused by unexpected obstacles

4.3 Travel distance

Figure 20 shows the average distance travelled at every trial. Although the path selection algorithm prefers longer path to shorter and risky ones, it can be seen that learning slightly decreases the average travelled distance. The most logical explanation is that manoeuvring around obstacles eventually increases the path length more than covering distances between obstacles. Again, like in case of replannings, it can be seen that suboptimal path are not worse than optimal ones.



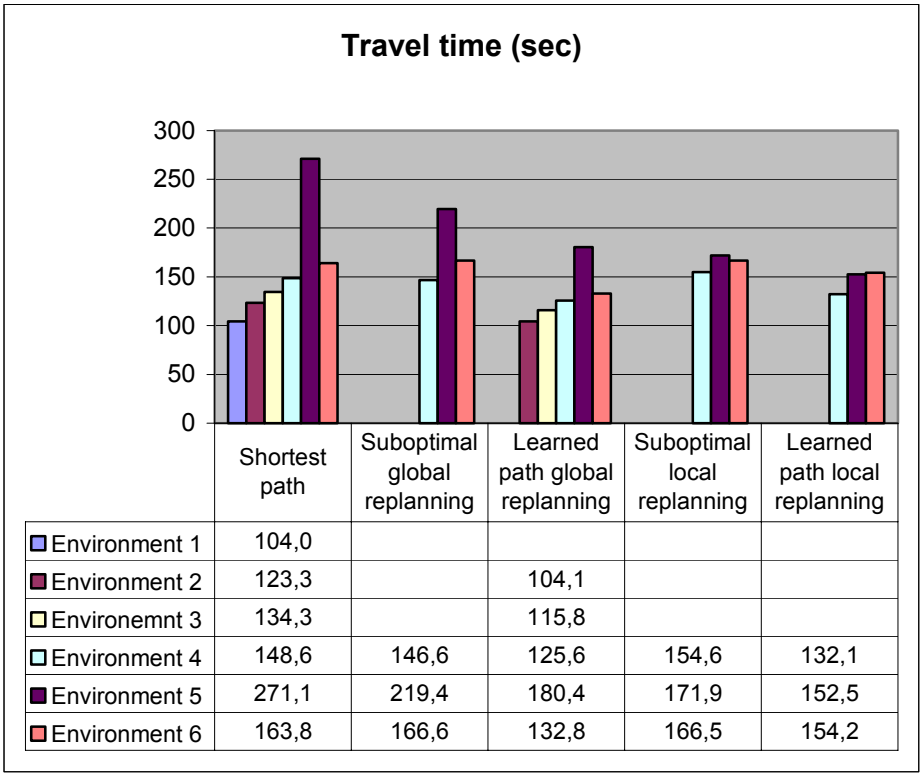
	Shortest path	Suboptimal global replanning	Learned path global replanning	Suboptimal local replanning	Learned path local replanning
Environment 1	2555,1				
Environment 2	2697,3		2584,6		
Environemnt 3	2768,1		2714,5		
Environment 4	2860,5	3100,6	2829,5	3234,9	2874,5
Environment 5	3478,5	3118,9	2982,6	3543,7	3218,7
Environment 6	3310,3	3349,2	2855,1	3437,4	3365,2

Figure 20. Average distance

4.4 Travel time

Figure 21 shows the average travel time of path following. The general conclusion about this data is the same as about the previous parameters. Learning improves the performance and modelling the environment decreases travel time. Again, following suboptimal path is about as time consuming as following the globally optimal one.

The very long travel time in the environment 1 with the shortest path algorithm seems somewhat anomalous. It is caused by few trials where the robot got trapped in a box canyon (see Figure 22). When trying to escape the canyon, the robot replanned always it counted an obstacle. Since it replanned globally, every replanning increased travel time. At the same time travel distance and deviation almost did not increase and therefore this anomaly is not visible on other charts.



	Shortest path	Suboptimal global replanning	Learned path global replanning	Suboptimal local replanning	Learned path local replanning
Environment 1	104,0				
Environment 2	123,3		104,1		
Environemnt 3	134,3		115,8		
Environment 4	148,6	146,6	125,6	154,6	132,1
Environment 5	271,1	219,4	180,4	171,9	152,5
Environment 6	163,8	166,6	132,8	166,5	154,2

Figure 21. Average travel time

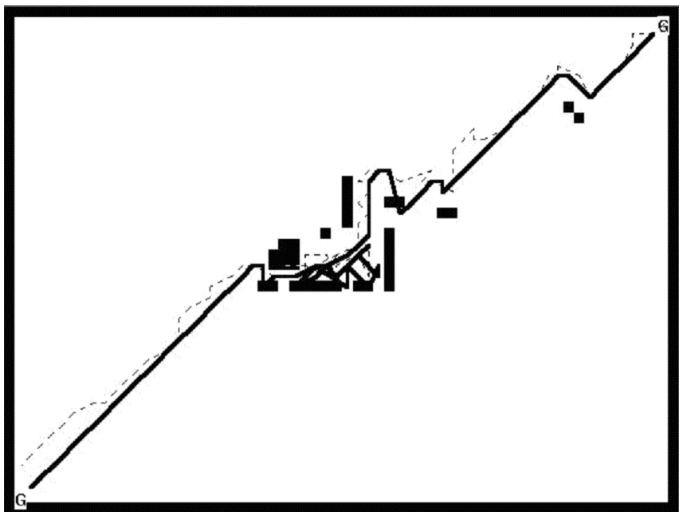
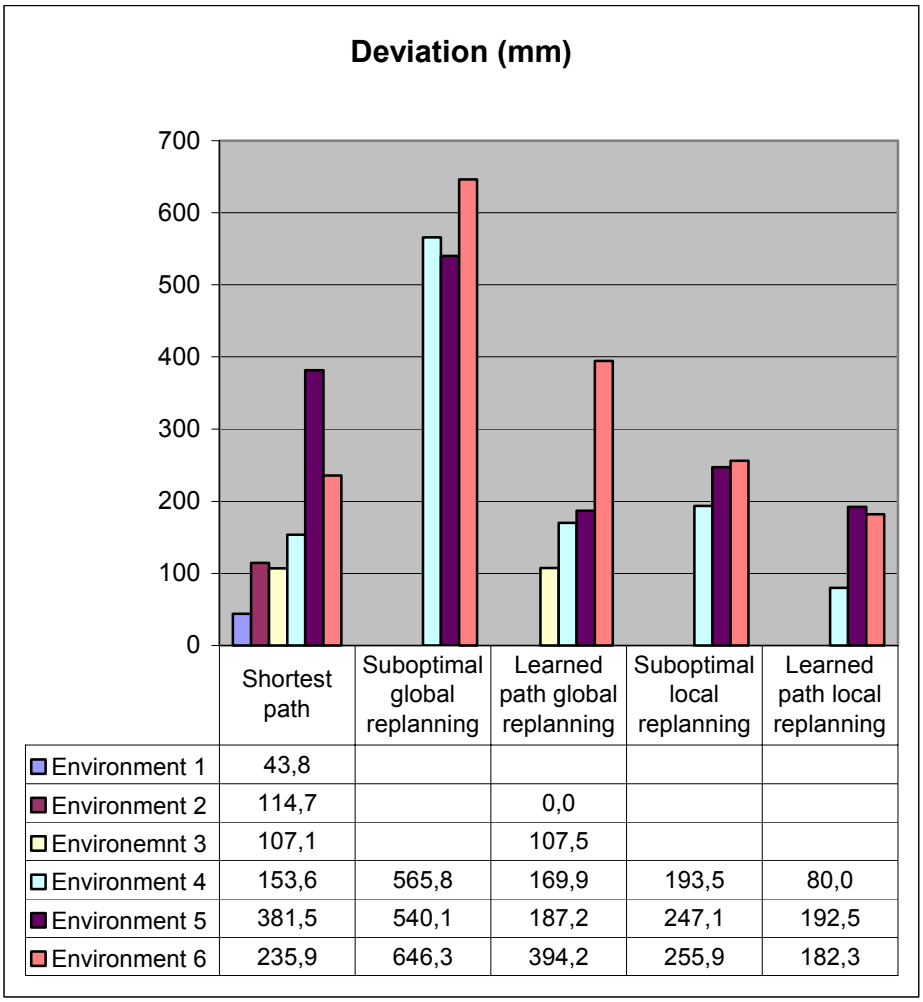


Figure 22. The robot escaping the box canyon

4.5 Deviation

Figure 23 shows the average deviation from the original path. Learning usually decreases deviation but it is not true in all cases. The deviation is large when the robot follows a suboptimal path and after counting an obstacle replans globally. This is not surprising since it might have initially driven in another direction and then switches to shortest path following.



	Shortest path	Suboptimal global replanning	Learned path global replanning	Suboptimal local replanning	Learned path local replanning
Environment 1	43,8				
Environment 2	114,7		0,0		
Environemnt 3	107,1		107,5		
Environment 4	153,6	565,8	169,9	193,5	80,0
Environment 5	381,5	540,1	187,2	247,1	192,5
Environment 6	235,9	646,3	394,2	255,9	182,3

Figure 23. Average deviation from the originally planned path

4.6 Learning

The experimental results represented above show that the path selection algorithm outperforms shortest path following and the learned routes are better than the shortest or suboptimal ones in terms of collision risk, travel time, distance and deviation. However, they do not prove that the performance increases because of learning and remembering.

Figure 24 shows how the performance of the system increases and the robot's behaviour stabilises during the test run. These tests are conducted in the Environment 4, a static unknown environment. Since the environment is static, the learning algorithm but not the environment causes any change in the robot's behaviour. Since the environment is unknown, it proves that the algorithm is able to learn even in the extreme case when very little information is available.

The path selection algorithm with both local and global replanning are verified against shortest path following. Both path selection algorithms stabilize rather quickly as they find and learn to use path that are better than the shortest one. The globally replanning path selection algorithm adapts to use a route with 0 replannings (Figure 25), while the path selection algorithm with local replanning adapts to use a more risky one (Figure 26). These results do not prove that globally replanning algorithm is better than the locally replanning algorithm because both of the algorithms look for routes that satisfy their criterion of safety defined by Equation 1. In this example, a route with couple of detected obstacles was considered to be good enough and the robot accepted the result. The globally replanning robot was fortunate to find a route with 0 obstacles and adapted to use this one.

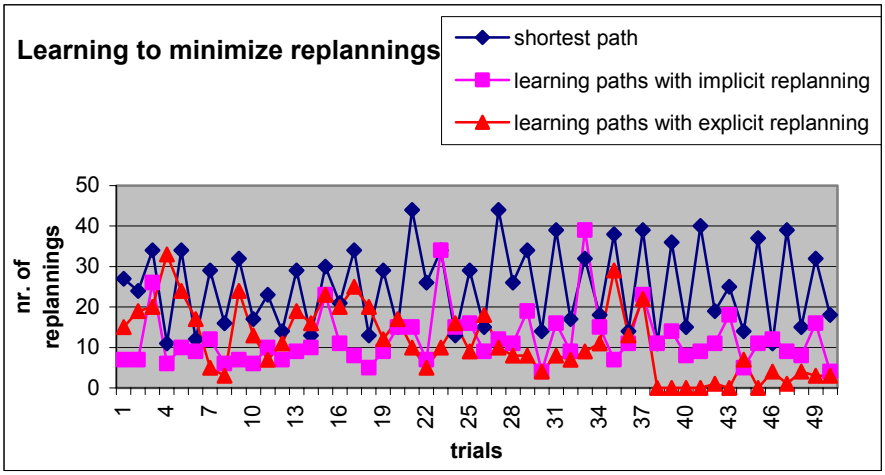


Figure 24. The learning curve of the path selection algorithm in the environment 4

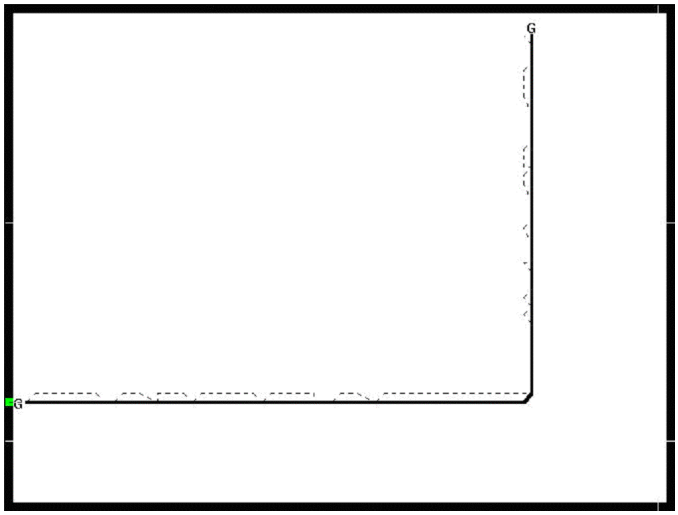


Figure 25. A learned route with the globally replanning path selection algorithm

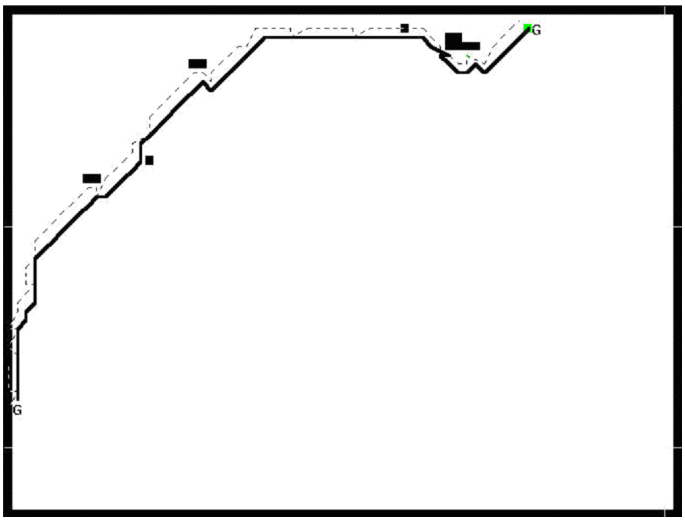


Figure 26. A learned route with the locally replanning path selection algorithm

4.7 Correlations

We here represent some more statistical data that helps to explain the experimental results. Table 9 represents the correlation coefficients between the number of replannings and other measured parameters. The data is given about the shortest path following in environments from 2 to 6. All these test runs contained 50 trials, which is sufficient for statistical analysis.

It appears that in all cases travel time and the number of replannings are highly correlated. This result is not surprising because the robot replans globally

using a wavetransform algorithm. For a reactively replanning robot the correlation could be weaker.

The correlation between the number of replannings and distance is weaker. This can be explained with that the robot does not have all global information available when it replans and therefore it does not always minimize the travel distance.

The correlation between the number of replannings and the deviation from the pre-planned path varies considerably from one test run to another. For example, while in environment 3 the correlation is highly negative; in environment 4 it is highly positive. It means that in some cases larger number of unknown obstacles causes a small deviation while a small number of unknown obstacles can cause a large deviation.

	Number of replannings				
Environment	2	3	4	5	6
Time	0,78	0,98	0,71	0,97	0,96
Distance	0,24	-0,06	0,45	0,63	0,82
Deviation	0,22	-0,67	0,65	-0,06	0,22

Table 9. Correlation coefficients for the shortest path following algorithm

5 Discussion

The most general conclusion about the experimental results is that in a partially modelled large-scale dynamic environment it is possible to optimize robot's behaviour by learning reliable trajectories.

Our initial hypothesis was that the traditional shortest path following strategy would soon outperform the innovative paths learning algorithm when the environment becomes known better. The chronological order of performing the test was in environments 5, 6, 4, 2, 1 and 3. The tests in a partially modelled environment were actually conducted later than in a unknown environment to show the limit where the path selection algorithm breaks down (it is also therefore the test in environments 2 and 3 do not include local replanning strategy because after conducting experiments in environments 4,5 and 6 it was not a research issue any more).

The test results did not confirm our initial hypothesis. On the contrary, it appeared that even if small obstacles are unmodelled the path selection algorithm improves the performance. The environment 2 depicted in Figure 11 is a good approximation of an unstructured human inhabited environment with unknown dynamic obstacles (people, vehicles, furniture or even sensor noise). From the experimental data it can be seen that the efficiency of the shortest path planning considerably decreases compared to the completely modelled environment 1.

Another unexpected conclusion is that global replanning does not improve the performance compared to the local replanning. We expected the global replanning strategy to perform better than the local one but the test results did not reveal any significant difference in performance.

Local replanning is performed mostly because sensorial capabilities and computational recourses are not sufficient for real-time global decision-making. Salich and Moreno describe in their paper the dilemma of authority vs. freedom [26]. The dilemma rises from that the global planner produces rigid orders while the local planner acts reactively, therefore the results are not globally optimal.

The test results reported here suggest that the dilemma of authority vs. freedom is irrelevant in partially unmodelled environments. The efficiency of path planning rather depends on the world model than of the path planning strategy. A global planner that does not have all global information available anyway fails to make a globally optimal plan and therefore the locally replanning agent performs equally well. From the test data it is obvious that global replanning does not give any particular advantage in terms of time, distance or collision risk. Moreover, local replanning should be preferred if the aim is to keep the robot close to its initially planned trajectory.

The next conclusion about the test data is that suboptimal path planning gives at least as good results as the optimal one. Intuitively one would expect an opposite conclusion. Speaking in decision theoretic terms, the globally replanning shortest path planner behaves as a utility maximizing rational agent. At every occasion it makes a decision that is best considering all global knowledge available. Therefore one would expect the cumulative utility to be higher. The suboptimal path planner (either globally or locally replanning) can be described as an explorative agent. It sometimes makes suboptimal decisions to explore the solution space and escape the local optimum. A reasonable explanation is that manoeuvring around obstacles takes significantly more recourses than traveling the distance between them.

The closer analysis of the test data reveals that among all the measured parameters, the robot's trajectory is the one that is most difficult to predict and control. It is difficult to foresee the extent of the deviation from the original track caused by an unknown object. Travel time and distance have a higher correlation to the number of replannings. Any changes in time, distance and deviation are caused by unexpected obstacles. We therefore can conclude that in order to optimize any other parameter (like time, distance, energy consumption, deviation) we have to minimize the number of unexpected events.

The general conclusion about the test data is that robot's behaviour is much more dependent on the environmental model than the path planning strategy. In order to optimize the behaviour of the robot, one should gain better knowledge about the environment. This can be achieved in two ways. The first option is to model the environment as closely as possible. The alternative way, reported in this paper, is to look for routes where uncertainty does not significantly influence the result.

6 Limits and further questions

Obviously these results cannot be interpreted as applying to all possible environments. They are limited to one randomly generated environment. If this environment is a good approximation of an unstructured, large, uncertain and dynamic working environment of mobile robots, it is likely that the results are also true in other cases. Repeating these tests in such an environment would give a confidence that the conclusions hold true in a large variety of scenarios. The problem rises from that a large, dynamic and uncertain environment is practically not controllable. The model environment represented here is on the contrary, fully controllable, and therefore we can be certain about the causes of the results.

These conclusions certainly cannot be generalised to topological maps and graph-based path planning since the models and methods considerably differ from those used for grid maps.

At the same time we suggest that the results can be generalised to other path planning methods than distance transform on a grid map. The performance of the shortest path planner and suboptimal path planner are not different. Therefore it is likely that the performance of the wavetransform planning or any other planning algorithm do not influence the performance either. Actually, any path planned on a grid map is suboptimal because the grid is digitized. It is possible that methods that use a varying resolution like [7] behave differently.

The approach of learning innovative tracks has two severe limitations. First, it assumes that localisation errors are small and do not accumulate. It is hard to predict how the localisation errors influence the result of path planning. Second, it can be applied only to missions where the robot repeatedly traverses between predefined target points. This assumption makes it possible to try several suboptimal routes and learn to use the most reliable ones.

7 Conclusions

This paper examined path planning strategies in large partially unknown environments. The robot learned innovative routes to find reliable trajectories and optimize robot's behaviour. The approach was verified to shortest path following in 6 different environments.

Experimental results lead to the following conclusions:

- The approach of learning and remembering reliable routes increases the performance of the robot.
- The behaviour of the robot is influenced by the knowledge it has about the environment but does not depend on the path planning strategy.
- To optimize travel time, distance, energy consumption, collision risk or deviation from the original path, the probability of unexpected events should be decreased as changes in the former parameters depend on the last one.

- Robot's trajectory in an uncertain environment is very difficult to predict and control because the deviation from the planned path is weakly correlated to the certainty of the model.

8 References

1. O.Khatib. Real-time obstacles avoidance for manipulators and mobile robots. *IEEE Int. Conf. On Robotics and Automation*, pages 500-505, March 1985.
2. B.Barraquand, J.Langlois, J.-C.Latombe. Numerical potential field technique for robot path planning. *Tech. Rep. Dept. of Computer Science, Report No. STAN-Cs-89-1285*, Stanford University, 1989.
3. R.Jarvis and K.Kang. A new approach to robot collision-free path planning. *Robots in Australia's Future Conference,* pages 71-79, 1986.
4. V.J.Lumelsky. A Comparative study of the path length performance of the maze-searching and robot motion planning algorithms. *IEEE Transactions on Robotics and Automation*,7(1):57-66, February 1991.
5. A.Stentz. The Focussed D* algorithm for real-time replanning. *Proc. of the Int. joint. Conf. Of Artificial Intelligence (IJCAI-95)*, August 1995.
6. F.Xu, H. van Brussel, M.Nuttin, R.Moreas. Concepts of Dynamic Obstacle Avodance and Their Extended Application in Underground Navigation. *Robotics and Autonomous Systems*, 42 :1-15, 2003.
7. A.Yahja, S.Singh, A.Stentz. An Efficient on-line Path Planner for Outdoor Mobile Robots. *Robotics and Autonomous Systems*, 32:129-143, Elsevier Science, 2000.
8. H.Seraji. New Traversability Indices and Traversability Grid for Integrated Sensor/Map-Based Navigation. *Journal of Robotic Systems*, 20(3):121-134, Wiley periodicals, 2003.
9. R.R.Murphy, K.Hughes, A.Marzilli and E.Noll. Integrating explicit path planning with reactive control of mobile robots using Trulla. *Robotics and Autonomous Systems*, 27(4):225-245, Elsevier Science, 1997.
10. S.Thrun, M.Bennewitz, W.Burgard, A.B.Cremers, F.Dellaert, D.Fox, D.Hähnel, C.Rosenberg, N.Roy, J.Schulte, and D.Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research (IJRR)*, 19(11), 2000.
11. N.Tomatis, G.Terrien, R.Piguet, D.Burnier, S.Bouabdallah and R.Siegwart. Design and System Integration for the Expo.02 Robot Workshop on Robots in Exhibitions, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, Switzerland, 2002.

12. U.Nehmzow. Quantitative analysis of robot–environment interaction– towards "scientific mobile robotics". *Robotics and Autonomous Systems*, 44(1) :55-68, 2003.

13. http://www.k-eam.com/robots/khepera/index.html

14. A. Zelinsky. Using Path Transforms to Guide the Search for Findpath in 2D. *The Int. Journal of Robotics Research,* 3(4):315-325, August 1994.

15. A.Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46-57, 1989.

16. A.Howard, H.Seraji. Vision-Based Terrain Characterization and Traversability Assessment. *Journal of Robotic Systems*, 18(10):577-587, Wiley periodicals, 2001.

17. D.B.Gennery. Traversability Analysis and Path Planning for Planetary Rovers. *Autonomous Robots*, 6:131-146, Kluwer, 1999.

18. U.Nehmzow, C.Owen. Robot Navigation in the Real World: Experiments with Manchester's Forty Two in Unmodified Large Environments. *Robotics and Autonomous Systems,* 33:223-242, Elsevier Science, 2000.

19. E.Kruse, F.M.Wahl. Camera-based Observation of Obstacle Motions to Derive Statistical Data for Mobile Robot Motion Planning. *Proc. Of IEEE Conference of Robotics and Automation,* 1:662-667, 1998.

20. E.Lemaster, S.Rock. A Local-Area GPS Pseudolite-Based Navigation System for Mars Rovers, *Autonomous Robots*, 14:209-224, 2003.

21. M.Kruusmaa, J.Willemson. Algorithmic Generation of path Fragment Covers for Mobile Robot Path Planning. *Technical Report*, 2003.

22. M. Kruusmaa, J.Willemson, K.Heero. Path Selection for Mobile Robots in Dynamic Environments. *Proc. of the 1st European Conference on Mobile Robots,* pages 113-118, Poland 2003.

23. K.Heero, J.Willemson, A.Aabloo, M.Kruusmaa. Robots Find a Better Way: A Learning Method for Mobile Robot Navigation in Partially Unknown Environments, submitted to IAS-8.

24. M. Kruusmaa. Repeated Path Planning for Mobile Robots in Uncertain Environments. *Proc. of the IASTED Int. Conf. Robotics and Automation*, pages 226-231, 2001.

25. R.Murphy, *Introduction to AI Robotics*, MIT Press 2000.

26. M.A.Salichs and L.Moreno. Navigation of Mobile Robots: Open Questions. *Robotica,* 18:227-234, Cambridge University Press, 2000.

APPENDIX D

K.Heero, A.Aabloo, M.Kruusmaa. On The Utility Of Exploration On Time-Critical Mobile Robots Missions. In *Proceedings of the 2nd European Conference on Mobile Robots (ECMR'05)*, pages 152-157, Ancona, Italy, September 2005.

ON THE UTILITY OF EXPLORATION ON TIME-CRITICAL MOBILE ROBOT MISSIONS

Kristo Heero

Data Security Laboratory
Cybernetica AS
Akadeemia tee 21, 12618 Tallinn
Estonia
kristo.heero@cyber.ee

Alvo Aabloo

Institute of Technology
Tartu University
Vanemuise 21, 51014 Tartu
Estonia
alvo.aabloo@ut.ee

Maarja Kruusmaa

Institute of Technology
Tartu University
Vanemuise 21, 51014 Tartu
Estonia
maarja.kruusmaa@ut.ee

ABSTRACT

This paper addresses the problem of the utility of exploration on time-critical mobile-robot missions. It is argued that in large environments mission-oriented mobile robot applications can become more efficient if the exploration strategy considers knowledge already gained and its applicability during the rest of the mission. This hypothesis is verified in a model test environment with Khepera robot. The conclusion is that mission-oriented exploration heuristics could be considered in mobile robot applications that are time-critical, where the robot is operating in a large unknown environment and if this environment is hazardous.

1. INTRODUCTION

A mobile robot operating in a real-world environment faces several fundamental problems. Most important of them is the ability to build the model of the environment [1], to plan routes [2] and to follow them while avoiding unknown obstacles [3].

These problems need to be tacked in all applications of mobile robots, such as transportation, surveillance or guidance [4, 5]. Vast amount of mobile robot research addresses the problem of exploration and environment mapping [6, 7]. Gaining knowledge about the surrounding environment and keeping it updated is the necessary precondition of successful performance.

In practical applications, exploration of the environment is usually risky and time-consuming. The environment can be hazardous and damage the robot. Exploration of a large environment takes lots of time and computational resources.

From the utilitarian point of view, knowledge about the environment is useful only as long as it increases the performance of the robot. In many cases the application does not require the exploration of the whole environment (e.q. fetch and carry tasks) while in other applications, such as demining or search and rescue, the task definition implies a systematic search of the whole area [8, 9, 10].

This paper addresses the problem of the utility of exploration for time-critical mobile robot missions. It is assumed that the environment is very large and therefore exploration is time-consuming. The environment can also be hazardous and degrade the performance of the robot or slow it down. The assumption is that exploration and mapping are not goals by itself but means that permit the robot to fulfill its mission.

Related work that consider the utility of route planning usually do not address the utility of exploration but rather evaluate the risk of navigation locally, e.g. in terms of possible collisions or the characteristics of the terrain [11, 12, 13, 14].

We propose a heuristic exploration strategy that chooses between exploring new areas and exploiting knowledge about the already explored areas. The decision-making is based on the mission plan. The heuristic decision maker takes into consideration the amount of knowledge acquired so far and its applicability during the rest of the mission.

We test the strategy in a model environment with the Khepera robot. The test results show that this mission-oriented heuristic can be useful for mobile robots on time-critical missions.

2. MISSION-ORIENTED EXPLORATION

In this section we describe the problem and outline the exploration strategy of the robot.

We assume that the robot is operating in a previously unknown environment. The goal of the robot is to fulfill a mission plan. The mission plan is known in advance, consisting of target points that the robot has to reach in a predefined order (e.g. a transportation task, escorting or surveillance problem).

The target points are defined by their global coordinates. This paper is concerned about exploration

strategy of time critical missions and therefore we do not address the problem of map building and localization in this context. We therefore assume that the robot is able to localize itself rather accurately (e.g. as with GPS or pseudolite localization). It is also assumed to have an environmental model in the form of a grid-based map. In the beginning of the mission the environment is unknown and the map contains only very general information (the size and shape of the environment). The robot has no knowledge about the obstacles or any other environmental factors that can degrade its performance. The robot learns the environment while traversing it and completing its mission. It updates the map when it detects obstacles with its on-board sensors.

2.1. Exploration Strategies

We verify two exploration strategies that are exploration oriented to a different extent. The bottom line of the first strategy is to always take a new route to the target (i.e. explore the environment) if it is expected to be better than the routes known so far.

The second strategy has a more conservative attitude against exploration. The difference from the first, greedy strategy is that, when gaining new knowledge it also considers the mission plan. The new knowledge is gained more probably when it is often used during the rest of the mission. Also knowledge that is used in the nearest future is gained more probably than knowledge used after a long time.

The robot has to reach predefined target points $G = \{g_1, g_2, \ldots, g_n\}$ where g is the grid cell on the map of the robot. The mission consists of traversing the target points in a predefined order $M = m_1, m_2, m_3, ..., m_i, m_{i+1}, ..., m_k$, where $m_j = (g_u, g_v)$ and $m_{j+1} = (g_v, g_w)$ for $1 \le j < k$, if $1 \le u, v, w \le n$.

In addition to the map that is constantly updated the robot also saves the entire followed path P. A path is stored as a sequence of grid cells. For every task $\mathrm{m_i}$ of the mission M it can choose between using an already followed path and a new path. The new path can contain segments that traverse unexplored regions.

The traversed paths are stored together with statistics characterizing their traversability. The average time of following a path $t(P)$ is used later when the strategy chooses between exploration and exploitation of the known tracks.

Every time the robot traverses the environment the map is updated so that the knowledge about the environment accumulates during the mission and every time when a new path is planned this new knowledge is taken into account.

2.2. Greedy Exploration Strategy

The greedy exploration strategy always chooses a new path if it predicts it to be better than the best known one. The predicted average time of the new path $t(P_{new})$ is verified to the best average time of the paths stored so far $t(P_{best})$. If $t(P_{new}) < t(P_{best})$ then the new path P_{new} is chosen.

While following the planned path, the robot does not try to stay on the predefined path if the obstacles are encountered but replans a new path to the target point through the possibly unexplored regions.

2.3. Conservative Exploration Strategy

This exploration strategy makes the decision between using the best-known path P_{best} and a new path P_{new} by considering exploitation of this knowledge in the future. It chooses P_{new} if the task m_i is not encountered often in the past and if it is needed often during the rest of the mission. The sooner during the mission new knowledge will be needed the more P_{new} is preferred.

Let

$$Past = \sum_{p=1}^{i-1} \begin{cases} 1, & m_p = m_i \\ 0, & \text{otherwise} \end{cases}$$

denote the number of similar tasks completed in the past. Since it was assumed that the robot knows its mission plan it can be counted how many times a task similar to m_i has to be completed in the future.

$$Future = 1 + \sum_{r=i+1}^{k} \begin{cases} \dfrac{\sum_{q=i+1}^{r} \begin{cases} 1, & m_q = m_i \\ 0, & \text{otherwise} \end{cases}}{r-i}, & m_r = m_i \\ 0, & \text{otherwise} \end{cases}$$

If $\dfrac{Past}{Future} > 1$, then it is determined that the robot has gained enough knowledge about the environment and the further exploration is not beneficial. The path P_{best} will be chosen and followed. If unexpected obstacles are encountered during the path following, the robot replans

the path but tries to return back to the initially chosen path P_{best} to avoid unexplored regions.

If $\frac{Past}{Future} \leq 1$ then the robot chooses P_{new} because very little of the environment is still explored or because the new knowledge gained can be used in the future to increase the performance.

3. EXPERIMENTAL SETUP

3.1. Test Environment

The experiments are conducted using a mini-robot Khepera. It is a differential drive miniature circular robot (with radius 26 mm) equipped with IR sensors for collision avoidance and it can be connected to a PC over a serial link.

The localization system is presented in Figure 1. A video camera is mounted to the ceiling to recognize the position and orientation of the robot. The PC processes the camera image to find robot's position and a computer algorithm controls the robot over a serial link. In this way localization errors are rather small (usually comparable to the size of the robot).

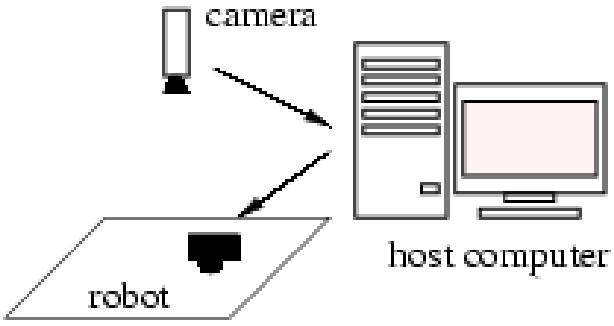


Figure 1. The experimental setup.

The test environment is represented in Figure 2. The size of the test environment is 2320mm × 1710mm.

Figure 3 shows the test environment as seen from the overhead camera. The position and the orientation of the robot are recognized with the help of 3 LEDs forming an equilateral rectangle.

Figure 4 is the graphical interface of the computer program that controls the robot and monitors its behavior. The coordinates of the target points are marked with the symbols G1, G2, etc. The thick line represents the path of the robot to the goal. Black cells represent obstacles detected with the onboard sensors. The gray and dark gray boxes are respectively unknown areas of slow and extra slow motion.

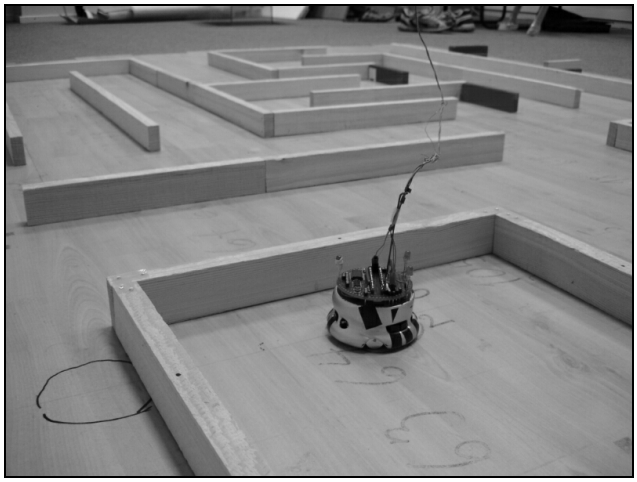


Figure 2. The test environment.

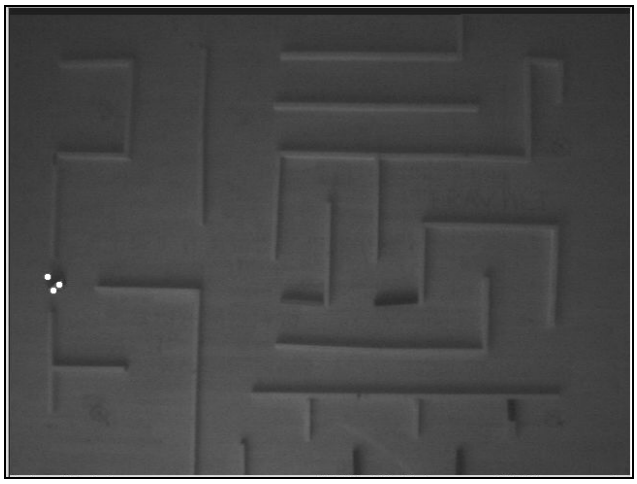


Figure 3. The test environment looked through the overview camera with the robot recognized by the 3 LEDs.

In the beginning of the mission the robot is not aware of any obstacles in the environment. While it traverses the environment it updates the map and records all the detected obstacles so that as the mission proceeds its environmental model becomes more and more complete and consistent.

In addition to the obstacles there are some regions in the environment where the motion of the robot is slowed down. These regions are introduced to simulate regions that in real robot applications are difficult to traverse (e.g. because of rough terrain). The robot is not aware of the presence and location of such regions and these are also

not reflected on the map. These areas are represented in the Figure 4 as the shaded regions.

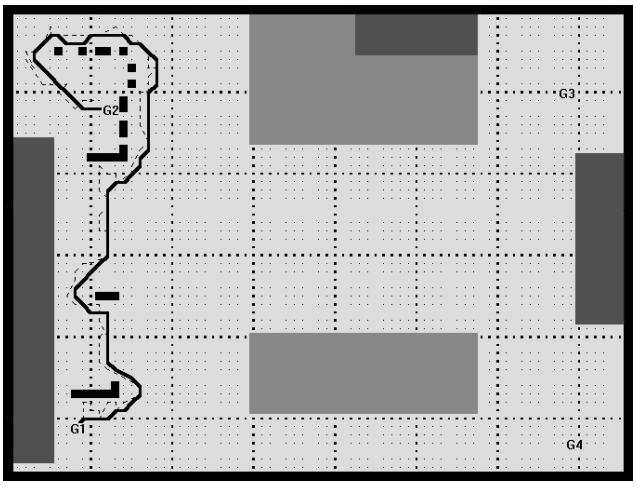


Figure 4. The control interface of the robot.

3.2. Test trials

The purpose of the experiments is to test whether the conservative exploration strategy gives better results than the greedy strategy in the presence of hazard and uncertainty.

We test both missions in equal conditions. The robot is given a mission consisting of 31 tasks and the goal of the robot is to fulfill the mission as fast as possible.

The environment, the task and the robot are identical in both cases. The test trials only differ by the exploration strategy used.

The mission plan (the sequence of target points to be traversed) is the following:

G1 G2 G1 G3 G1 G2 G4 G2 G1 G2 G1 G3 G1 G2 G4 G2 G1 G2 G1 G3 G1 G2 G4 G2 G1 G2 G1 G3 G1 G2 G4 G2

The target point G1 corresponds to the point marked with G1 in Figure 4 in the lower-left corner, G2 in the upper-left corner, etc. in the clockwise direction. This plan implies that the robot traverses often between G1 and G2 but quite seldom between the target points G2 and G4 or G1 and G3. Some parts of the environment, like those between G4 and G3 are not traversed at all.

Since the different regions of the map are traversed with the different intensity, exploring some regions becomes more important from the point of view of the mission than exploration of other regions.

The model environment is kept static during the mission. No dynamic obstacles are introduced. Obviously, this is not a realistic assumption on real robot missions. However, the goal of the tests is to show the advantages or disadvantages of an exploration strategy and the static environment guarantees that if one exploration strategy outperforms another then this is caused by the strategy but not by the changes in the environment.

Although the environment is kept static, the robot in the model environment still has to tackle problems caused by uncertainty of sensor readings, odometric errors and small localization errors due to the image recognition system. The generation of new paths P_{new} is stochastic and therefore the performance of the robot depends to a great extent on a stochastic algorithm. We therefore conducted several pairs of test trials to show how much the test results diverge.

The map and the memory (containing the traversed paths and their average traveling time) are stored after every task. All test data is available at *http://math.ut.ee/~kristo/khepera/heuristic/*.

4. EXPERIMENTAL RESULTS

The goal of the robot was to fulfill the mission as fast as possible. Therefore the average time of the mission is the most important parameter indicating the efficiency of the exploration strategy.

The chart in Figure 5 shows the duration of the mission. Time of fulfilling 31 tasks is 16.74% shorter when the conservative exploration strategy is used. While both of the exploration strategies perform equally well in the beginning of the mission, the conservative learning strategy starts outperforming the greedy strategy after 10 first tasks.

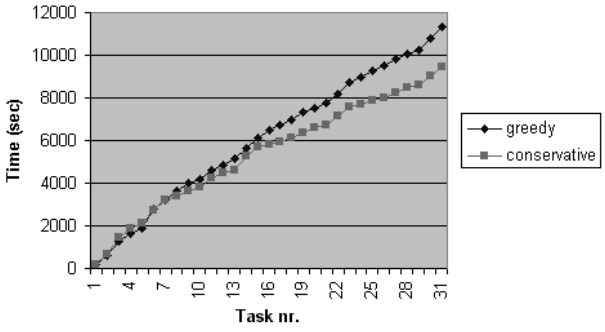


Figure 5. The duration of the mission.

The white areas in Figure 6 show regions explored during the mission when the conservative exploration strategy is used. The mission plan requested frequent traversing between the lower left and upper left corner of the

environment and it appears that the robot has explored these regions most extensively while the rest of the environment is searched less thoroughly.

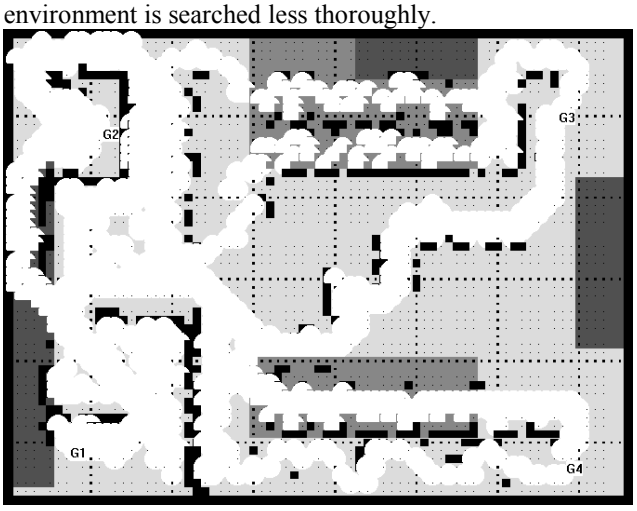


Figure 6. Explored areas with the conservative exploration strategy.

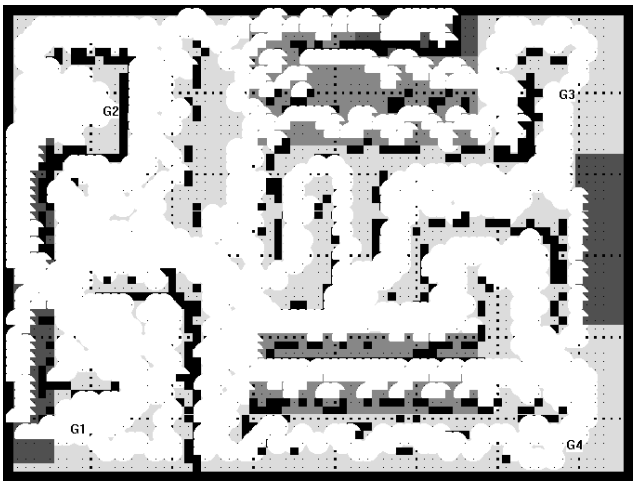


Figure 7. Explored areas with the greedy exploration strategy.

The white areas in Figure 7 show regions explored with the greedy exploration strategy. It appears that almost the whole environment is searched and most of the obstacles are mapped. The robot has also been extensively traversing the areas of slow motion (marked with gray) that have eventually slowed the mission down as well as frequent replanning and maneuvering around detected obstacles.

Another indicator that shows the difference between the conservative and the greedy exploration algorithm is the number of reused paths. Traversed paths are stored in robot's memory together with statistics showing their average traversing time. Results show that the greedily exploring robot used already traversed paths in 51% of tasks while the conservative robot relied on its past experiences in 67% of the cases.

5. CONCLUSIONS

This paper presented an exploration strategy for a mobile robot in large hazardous environments. It was presumed that the robot is working under time constraints. We presented a heuristic exploration strategy that chooses between exploration and exploitation considering the amount of knowledge gained so far and the applicability of this knowledge during the rest of the mission.

We simulated a time-critical mission by conducting experiments in a model environment with a Khepera and verified the heuristics with a greedy exploration strategy. The test results showed that the robot using the conservative exploration strategy is able to fulfill the mission approximately 17% faster than the robot using the greedy exploration strategy.

These test results reveal that it can be useful to choose between exploration of the environment and the exploitation of the knowledge gained depending of the nature of the mission and the environment. Gaining as much knowledge as possible about the surrounding is not necessary beneficial if the mission time is limited.

The performance of the conservative exploration strategy undoubtedly depends on the environment where the robot is operating and on the mission assigned. We therefore are careful with generalizing these results too much. Certainly, there exist environments and assignments where the greedy exploration strategy would be more efficient. More experiments in different environments (e.g. cluttered, free space, corridor-environments, multiple rooms, etc.) and different path planning methods are required to validate the presented approach.

We conclude that mission-oriented exploration heuristics could be considered in mobile robot applications that are time-critical, where the robot is operation in a large unknown environment and when this environment is dangerous. We also suggest that this or similar heuristics can be applied for other learning problems in mobile robotics.

ACKNOWLEDGEMENTS

This paper is supported by Estonian Science Foundation grant No. 5613, and Tiger Leap Program of The Estonian Information Technology Foundation.

6. REFERENCES

[1] Thrun, S., Burgard, W., and Fox, D. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning* Vol. 31, pp. 29-53, 1998.

[2] D. Ferguson and A. Stentz, "The Delayed D* Algorithm for Efficient Path Replanning." *Proc. of the 2005 IEEE Int. Conf. on Robotic and Automation, (ICRA 2005),Barcelona, Spain, April, 18-22.*

[3] J. Latombe, "Robot Motion Planning." Kluwer Academic Publishers, Boston, MA, 1991.

[4] B.Brumitt, A. Stentz, M. Herbert and The CMU UGV Group, "Autonomous Driving with Concurrent Goals and Multiple Vehicles: Mission Planning and Architecture," *Autonomous Robots*, Vol. 11, Kluwer Academic Publishers, pp. 103-115, 2001.

[5] B.Brumitt, A. Stentz, M. Herbert and The CMU UGV Group, "Autonomous Driving with Concurrent Goals and Multiple Vehicles: Experiments and Mobility Components, *Autonomous Robots,* Vol. 12, Kluwer Academic Publishers, pp. 135-156, 2002.

[6] Thrun, S. Robotics mapping: A survey. Tech. Rep. CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213, February 2002.

[7] F. Amigoni and A. Gallo, "A Multi-Objective Exploration Strategy for Mobile Robots." *Proc. of the 2005 IEEE Int. Conf. on Robotic and Automation, (ICRA 2005),Barcelona, Spain, April, 18-22.*

[8] E. Garcia and P. Gonzalez de Santos, "Mobile-robot navigation with complete coverage of unstructured environments," *Robotics and Autonomous Systems*, Vol. 46, Elsevier Science, pp. 195-204, 2004.

[9] H. Choset, "Coverage of Known Spaces: The Boustrophedon Cellular Decomposition," *Autonomous Robots*, Vol. 9, Kluwer Academic Publishers, pp. 247-253, 2000.

[10] A. Poncela, E. J Perez and A. Bandera, C. Urdiales, F. Sandoval, "Efficient integration of metric and topological maps for directed exploration of unknown environments," *Robotics and Autonomous Systems,* Vol. 41, pp. 21-39, 2002.

[11] R. Manduchi, "Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation," *Autonomous Robots*, Vol. 18, Kluwer Academic Publishers, pp. 81-102, 2005.

[12] T. Belker, M. Hammel and J. Hertzberg, „Learning to Optimize Mobile Robot Navigation Based in HTN Plans," *Proc. Of the 2003 IEEE Int. Conf. on Robotic and Automation, (ICRA 2003),* Taipei, Taiwan Sept. 14-19, pp. 4136 – 4141, 2003.

[13] Z. Shung and J. Reif, „On Energy-minimizing Path on Tarrains for a Mobile Robot," *Proc. of the 2003 IEEE Int. Conf. on Robotic and Automation, (ICRA 2003),* Taipei, Taiwan Sept. 14-19, pp. 3782 – 3788, 2003.

[14] R. Philippsen and R. Siegwart, "An Interpolated Dynamic Navigation Function." *Proc. of the 2005 IEEE Int. Conf. on Robotic and Automation, (ICRA 2005),Barcelona, Spain, April, 18-22.*

APPENDIX E

Detailed Description of the Test Environments

Introduction

This Appendix describes in detail all test environments used in Appendices A, B, C, and D. There are 7 different environments. In Appendix A is used environment 5; in Appendix B is used environments 1, 2, and 3; in Appendix C is used environments 1, 2, 3, 4, 5, and 6; in Appendix D is used environment 7.

Common in Environments 1 - 6

The size of the environment is 1860×1390 mm. Example of the real environment is represented in Figure 1. It is divided into 20×15 cells indexed with numbers 1-300 (started from upper-left cell and propagated from left to right) for the placement of obstacles. Shapes and amount of the obstacles are represented in Table 1.

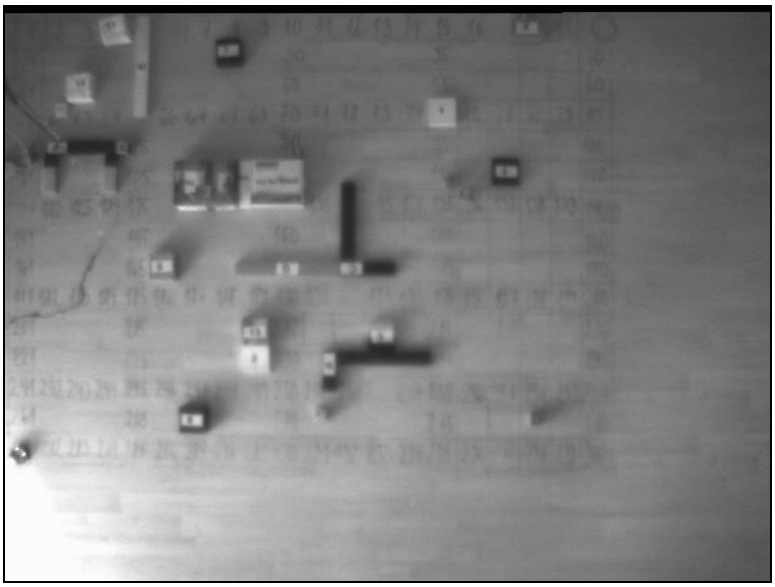


Figure 1. Example of the real environment

Table 1. Shapes and amount of obstacles

	I-shapes	L-shapes	C-shapes	Rect. 1	Rect. 2	Rect. 3	Rect. 4
Amount	2	2	1	1	1	10	8
Size (mm)	320×40	120×320 & 160×280	160×240×160	140×385	100×105	80×80	40×40

All locations of the obstacles and orientations (N, W, S, E) of the unsymmetrical obstacles (I-shapes, L-shapes, C-Shapes, rec. 1, and rect. 2) on a placement grid are generated using a random number function.

Obstacles of type I-shapes, L-shapes, C-Shapes, rec. 1, and rect. 2 are set down on the placement grid if randomly chosen cell and necessary adjacent cells are free, otherwise next cell is randomly chosen.

Obstacles of type Rect. 3 are set down on the placement grid if randomly chosen cell is free, otherwise new cell is randomly chosen.

Obstacles of type Rect. 4 are set down on the placement grid only if randomly chosen cell is free.

Randomly generated poses of the obstacles in the environments 1-4, 5, and 6 are represented in Table 2, Table 3, and Table 4 respectively.

Table 2. Poses of the obstacles in environments 1 – 4

Type	Cell Index (and Orientation)
Unsym.	83-1; 233-4; 152-3; 170-1; 25-4; 18-1; 109-2
Rect. 3	75; 229; 213; 117; 166; 267; 28; 4; 209; 43; 196; 241; 104; 247; 168; 146; 199; 189; 200; 126
Rect. 4	62; 81; 85; 91; 115; 136; 271; 278; 45; 274; 126; 77; 185; 36; 80; 2

Table 3. Poses of the obstacles in environments 5

Type	Cell Index (and Orientation)
Unsym.	83-1; 233-4; 152-3; 170-1; 25-4; 18-1; 109-2
Rect. 3	154; 73; 107; 77; 257; 199; 134; 266; 143; 188; 275; 291; 3
Rect. 4	1. 287;197;274;211;71;191;230;275;223;78;238;71;104;145 2. 253;120;134;234;9;3;288;225;299;90;36;111;79;234 3. 206;291;154;1;237;110;66;71;295;81;174;107;221;143 4. 31;210;296;253;228;59;266;183;49;152;142;30;180;156 5. 97;93;113;282;281;127;113;77;282;246;68;261;240;160 6. 290;207;62;233;197;283;224;149;245;146;21;158;28;133 7. 52;108;22;107;289;20;194;229;232;121;244;102;286;185 8. 116;153;74;4;123;215;6;239;238;63;39;58;4;163 9. 131;137;87;201;227;207;52;243;90;34;278;285;100;221 10. 175;68;50;4;53;287;45;138;277;190;249;250;262;197 11. 274;255;260;13;63;296;73;285;60;124;54;84;113;170 12. 28;34;12;125;293;88;206;277;34;276;190;230;19;14 13. 20;251;89;105;48;157;132;19;117;106;146;283;227;2 14. 272;173;4;63;19;230;180;173;244;98;123;274;222;7 15. 27;299;270;291;283;125;146;77;63;131;94;29;229;138 16. 173;67;291;33;210;274;246;144;233;100;297;158;134;156 17. 4;62;189;236;60;108;15;245;138;251;270;76;208;291 18. 184;11;269;256;32;79;197;154;129;274;53;140;62;43 19. 143;214;228;141;285;207;118;191;278;68;197;124;117;5 20. 10;36;153;195;295;55;40;223;209;135;276;48;154;181 21. 77;276;107;128;183;225;50;54;137;269;200;275;152;36 22. 116;298;88;159;120;164;255;53;182;73;181;300;2;21 23. 41;48;264;126;2;73;80;99;63;123;186;18;166;6 24. 35;259;140;218;134;57;152;184;167;37;54;116;58;101 25. 65;42;214;96;293;231;2;59;46;149;1;268;118;183 26. 178;279;13;273;134;189;182;264;173;15;46;46;26;106 27. 76;86;297;292;199;104;161;90;114;121;268;56;63;119 28. 199;23;164;131;29;2;151;22;116;80;5;173;11;6 29. 39;209;180;193;28;10;231;27;125;135;67;6;172;192 30. 20;12;140;96;114;221;102;36;200;278;90;203;211;221 31. 18;290;189;286;37;134;224;91;5;282;252;3;219;282 32. 165;119;20;60;180;244;300;88;94;192;219;18;114;216 33. 172;159;56;273;116;108;206;36;180;76;60;166;184;145 34. 11;142;190;290;100;74;193;293;24;292;199;1;197;138 35. 283;240;251;249;75;35;198;167;276;65;18;55;240;192

	36. 202;156;228;86;218;189;284;99;122;89;113;1;191;118 37. 28;238;189;231;150;14;52;32;11;168;296;108;263;46 38. 45;65;225;231;271;71;184;40;243;7;80;147;193;55 39. 185;145;278;104;212;198;148;18;58;199;15;240;127;196 40. 16;270;224;285;263;210;79;142;198;191;182;288;216;186 41. 291;169;286;133;298;231;41;157;210;77;160;216;31;200 42. 201;172;202;251;232;203;90;162;208;158;277;69;224;170 43. 138;261;125;232;65;96;79;157;128;111;298;2;17;269 44. 19;13;176;69;227;147;251;64;31;25;178;2;176;295 45. 31;55;298;232;51;33;118;79;284;57;163;232;30;286 46. 222;89;161;164;130;72;86;206;170;8;225;161;7;129 47. 288;179;21;231;286;37;81;276;7;27;203;77;265;77 48. 237;135;150;102;49;89;204;249;8;124;36;156;269;19 49. 7;66;163;276;5;213;59;299;123;151;141;169;173;113 50. 246;102;159;79;10;217;113;152;173;94;244;7;75;174

Table 4. Poses of the obstacles in environments 6

Type	Cell Index (and Orientation)
Unsym.	83-1; 233-4; 152-3; 170-1; 25-4; 18-1; 109-2
Rect. 3	1. 236;254;77;14;270;243;13;244;97;103;82;121;292;57;109;9;104;31;160;155 2. 236;254;77;122;270;262;13;146;97;103;82;121;292;141;109;89;104;4;160;155 3. 236;134;77;122;270;262;43;146;97;103;82;59;292;141;109;89;196;4;160;155 4. 236;134;77;122;270;91;43;146;97;103;82;59;292;141;109;45;196;4;160;155 5. 236;60;228;122;270;91;240;146;239;103;82;141;281;141;109;45;164;4;299;155 6. 236;60;228;122;77;91;240;146;239;103;82;141;281;141;130;45;164;4;299;155 7. 236;60;228;122;86;91;240;182;196;18;82;141;281;141;258;45;164;36;168;160 8. 236;60;206;20;86;91;240;182;196;18;82;141;98;271;258;45;164;36;168;160 9. 139;60;9;67;86;91;240;182;196;179;213;141;209;113;258;45;164;36;168;69 10. 139;60;9;155;24;91;240;182;196;179;213;141;209;9;206;45;164;36;168;69 11. 194;60;9;288;66;261;202;55;196;179;129;141;209;257;72;145;296;82;168;69 12. 194;60;134;288;249;261;202;240;196;246;129;141;279;257;184;145;296;82;168;296 13. 194;60;134;28;249;179;202;240;196;224;129;141;279;190;184;299;296;82;168;185 14. 194;60;5;28;207;179;202;240;196;224;129;141;196;190;76;299;296;82;168;185 15. 194;60;5;28;167;216;202;240;196;224;129;141;196;190;116;37;296;82;168;185 16. 75;178;200;28;167;216;202;240;196;224;134;52;294;190;116;37;296;82;168;185 17. 75;178;200;28;167;216;202;240;196;224;134;52;294;190;116;37;296;82;168;185 18. 75;176;200;28;229;216;202;62;196;224;134;270;294;190;118;37;296;177;168;185 19. 75;176;27;75;259;216;202;62;196;224;134;270;114;135;47;37;296;177;168;185 20. 116;176;27;75;259;216;202;62;196;224;126;270;114;135;47;37;296;177;168;185 21. 116;176;27;75;259;216;202;283;196;224;126;270;114;135;47;37;296;198;168;185 22. 116;176;27;75;259;127;202;283;196;224;126;270;114;135;47;21;296;198;168;185 23. 241;170;27;75;118;76;103;283;272;224;254;73;114;135;267;292;96;198;268;185 24. 241;170;206;75;118;76;103;93;180;224;254;73;204;135;267;292;96;36;80;185 25. 241;170;206;75;118;76;103;184;180;13;254;73;204;135;267;292;96;10;80;236 26. 191;170;56;75;118;76;99;117;180;13;7;73;103;135;267;292;107;60;80;236 27. 191;170;33;75;118;76;99;117;180;13;7;73;166;135;267;292;107;60;80;236 28. 57;170;33;75;118;76;99;117;180;13;291;73;166;135;267;292;107;60;80;236 29. 57;120;33;75;118;76;15;197;180;274;291;156;166;135;267;292;127;1;80;123 30. 57;120;252;75;118;76;15;197;180;274;291;156;186;135;267;292;127;1;80;123 31. 131;120;252;75;118;76;15;197;180;168;178;156;186;135;267;292;127;1;80;276 32. 131;120;252;279;293;76;15;197;180;168;178;156;186;70;174;292;127;1;80;276 33. 131;32;252;279;293;76;15;197;180;168;178;237;186;70;174;292;127;1;80;276 34. 147;32;252;279;293;76;15;197;180;168;168;237;186;70;174;292;127;1;80;276 35. 147;32;252;279;74;76;298;197;129;168;168;237;186;70;88;292;79;1;238;276 36. 147;32;252;279;74;76;298;197;129;168;168;237;186;70;88;292;79;1;238;276 37. 147;32;252;279;74;76;298;197;129;168;168;237;186;70;88;292;79;1;238;276 38. 146;208;252;279;32;226;298;197;129;168;50;79;186;70;45;1;79;1;238;276 39. 146;208;252;279;32;226;99;197;129;71;50;79;186;70;45;1;259;1;238;50 40. 261;53;252;279;32;226;99;197;248;71;172;125;186;70;45;1;259;1;260;50 41. 50;53;252;279;32;226;99;197;248;71;215;125;186;70;45;1;259;1;260;50

	42. 50;53;95;279;32;226;99;197;248;71;215;125;242;70;45;1;259;1;260;50 43. 50;53;95;279;32;226;99;197;243;71;215;125;242;70;45;1;259;1;85;50 44. 50;14;95;279;32;226;115;197;243;71;215;253;242;70;45;1;253;1;85;50 45. 50;207;228;12;32;226;115;197;243;235;215;251;106;54;45;1;253;1;85;23 46. 50;207;228;117;32;226;146;100;243;235;215;251;106;21;45;1;271;209;85;23 47. 50;207;44;117;32;226;240;100;243;235;215;251;93;21;45;1;185;209;85;23 48. 137;207;44;117;18;226;240;100;243;199;15;251;93;21;174;1;185;209;85;257 49. 137;207;44;117;159;226;240;154;243;110;15;251;93;21;245;1;185;159;85;210 50. 109;207;235;117;159;226;240;158;243;110;45;251;121;21;245;1;185;39;85;210
Rect. 4	1. 5;43;50;73;167;187;256;295;114;263;60;16;83;53;235;243; 2. 15;16;51;60;115;148;183;194;12;178;277;68;57;30;206;250; 3. 12;49;50;93;129;207;269;270;47;173;271;233;257;265;268;191; 4. 9;141;142;156;215;245;279;280;97;15;250;175;177;267;30;13; 5. 3;44;79;146;172;241;289;292;262;104;109;53;239;7;234;85; 6. 19;69;100;112;177;187;214;258;280;47;200;64;5;42;201;108; 7. 10;21;83;87;124;226;249;288;168;205;243;74;287;295;172;27; 8. 9;33;37;80;249;252;291;297;187;111;134;185;66;300;93;232; 9. 80;90;131;163;233;240;254;300;160;22;10;227;103;76;132;12; 10. 33;155;193;204;208;217;241;276;274;264;70;89;47;118;255;103; 11. 52;72;81;162;183;199;210;259;127;7;205;255;219;281;89;220; 12. 42;57;88;108;127;151;184;228;252;218;285;253;59;40;290;135; 13. 1;62;79;107;123;268;289;298;105;57;96;128;18;12;112;54; 14. 102;109;122;146;192;214;272;283;237;143;32;249;243;2;255;92; 15. 34;145;150;201;219;222;235;284;149;296;217;186;196;218;22;117; 16. 64;151;162;163;188;212;230;249;233;178;169;60;184;289;103;285; 17. 27;54;59;85;102;125;164;190;298;42;93;26;103;219;138;90; 18. 53;62;73;92;117;175;275;296;227;232;249;199;259;16;86;188; 19. 5;11;24;40;65;182;280;283;287;144;197;120;168;199;202;239; 20. 76;83;114;143;162;183;189;231;171;111;175;244;219;233;220;19; 21. 1;99;136;146;164;200;229;294;215;249;110;105;51;219;61;10; 22. 53;59;100;120;223;234;274;279;43;10;280;2;27;210;199;145; 23. 1;35;80;99;115;173;239;283;65;113;15;188;215;163;158;272; 24. 116;136;218;260;267;282;291;298;169;154;56;100;127;248;204;75; 25. 96;112;171;188;193;214;228;261;292;51;273;62;141;53;130;184; 26. 43;135;142;158;196;210;266;267;279;250;100;118;167;216;184;247; 27. 106;146;168;193;199;252;277;295;210;176;51;27;227;204;140;55; 28. 92;93;169;190;201;227;230;240;38;249;67;112;41;137;72;211; 29. 21;44;100;129;180;217;220;284;248;112;288;46;111;41;115;223; 30. 3;127;165;209;244;281;295;299;182;46;296;289;71;157;37;43; 31. 49;65;66;90;105;107;125;213;291;109;220;91;129;252;285;5; 32. 29;116;129;157;185;218;271;285;73;63;222;240;204;43;107;183; 33. 18;41;58;68;159;241;242;260;240;233;75;171;269;33;62;218; 34. 46;74;118;168;203;239;251;283;146;270;134;212;145;3;137;64; 35. 10;46;58;98;157;206;256;281;284;40;283;290;175;234;19;205; 36. 56;91;123;158;206;269;288;292;207;213;150;159;282;153;215;120; 37. 4;50;67;81;141;187;212;229;215;224;39;120;189;68;73;243; 38. 58;109;123;128;158;194;214;225;248;247;164;238;229;16;20;165; 39. 42;92;100;123;142;162;164;168;246;30;267;184;47;70;206;121; 40. 94;122;184;201;209;246;274;285;81;26;147;50;120;121;12;235; 41. 47;64;70;73;92;102;190;290;156;224;267;136;130;57;62;208; 42. 67;76;159;177;197;285;287;299;215;298;114;81;194;217;284;210; 43. 10;15;123;130;189;233;244;283;84;99;289;215;96;154;73;29; 44. 27;37;64;87;139;141;185;261;107;194;217;253;158;249;294;6; 45. 11;58;99;106;128;139;256;294;38;31;116;44;33;288;204;262; 46. 58;115;134;162;203;242;249;297;57;292;151;76;53;132;81;176; 47. 35;42;99;123;149;241;266;300;16;291;260;273;249;76;148;143; 48. 62;76;132;142;170;207;242;274;48;211;113;291;265;71;94;104; 49. 1;14;133;218;234;252;260;275;157;222;208;43;256;37;91;110; 50. 43;130;216;246;247;260;284;285;108;171;76;110;69;90;148;141;

Environment 1 - static, known

All obstacles are static and known *a priori* (shown in Figure 2).

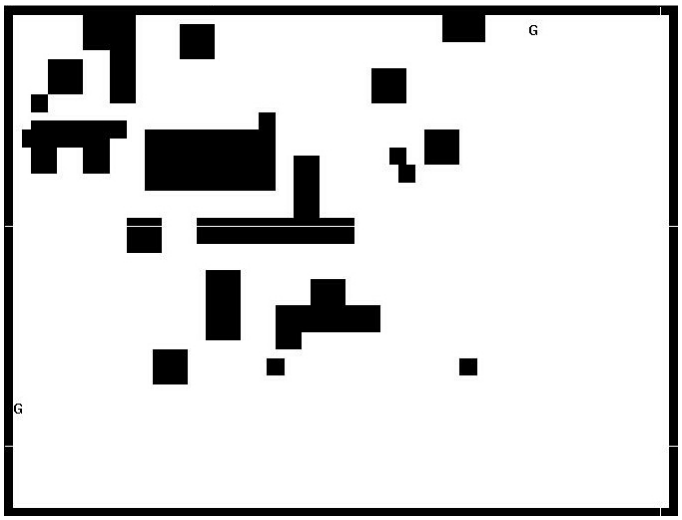


Figure 2. *A priori* map

Environment 2 - static, large obstacles known

All obstacles are static. Only large obstacles of type I-shapes, L-shapes, C-Shapes, rec. 1, and rect. 2 are known *a priori* (shown in Figure 3).

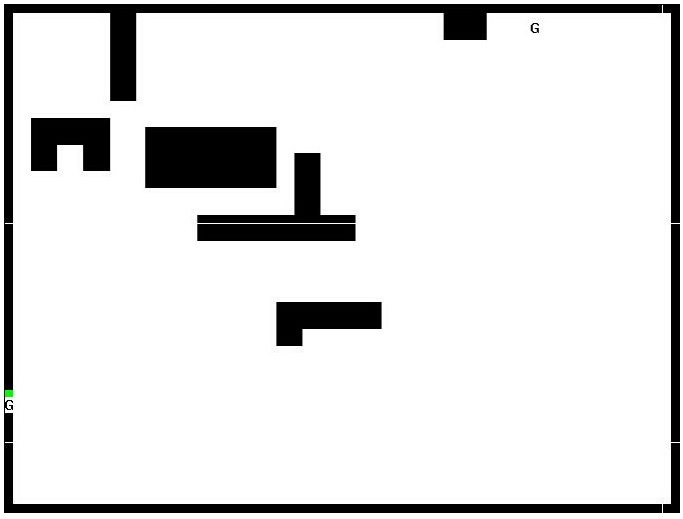


Figure 3. *A priori* map

Environment 3 - static, small obstacles known

All obstacles are static. Only small obstacles of type Rect. 3 and Rect. 4 are known *a priori* (shown in Figure 4).

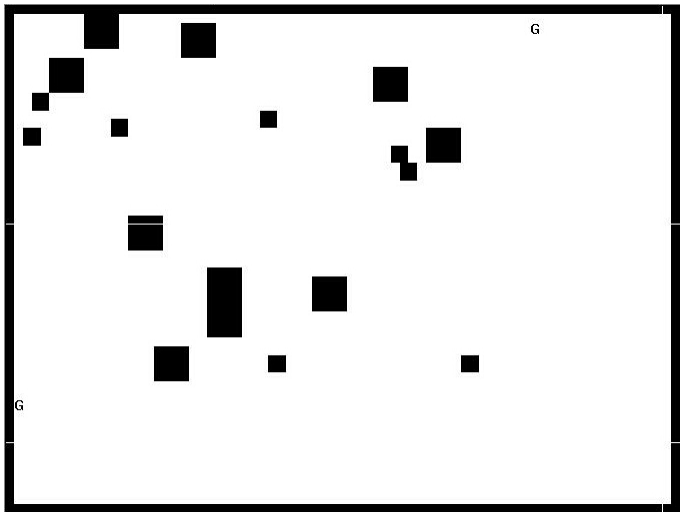


Figure 4. *A priori* map

Environment 4 - static, unknown

All obstacles are static and unknown *a priori* (shown in Figure 5).

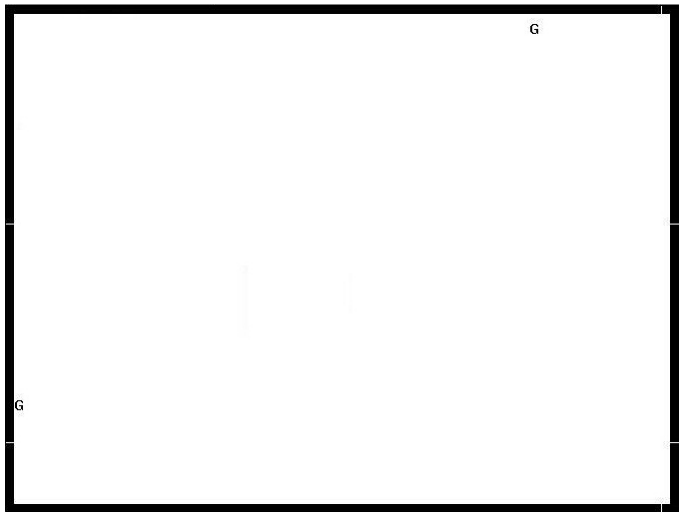


Figure 5. *A priori* map

Environment 5 - slightly dynamic, unknown

All obstacles are unknown *a priori* (shown in Figure 6). Only small obstacles of type Rect. 4 are dynamic i.e. after every traversal they are randomly replaced.

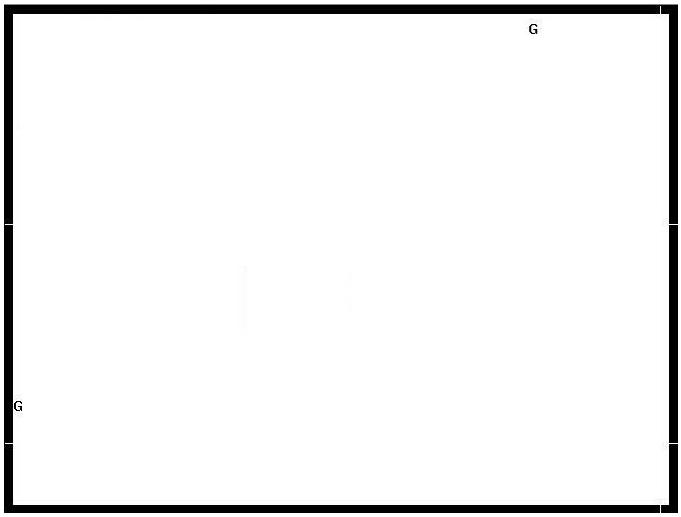


Figure 6. *A priori* map

Environment 6 - moderately dynamic, unknown

All obstacles are unknown *a priori* (shown in Figure 7). Small obstacles of type Rect. 3 and Rect. 4 are dynamic i.e. after every traversal they are randomly replaced with probability 0.2 and 1 respectively. Additionally to obstacle placement rules, the obstacle of type Rect. 3 is not put down on the grid if randomly chosen cell is already occupied.

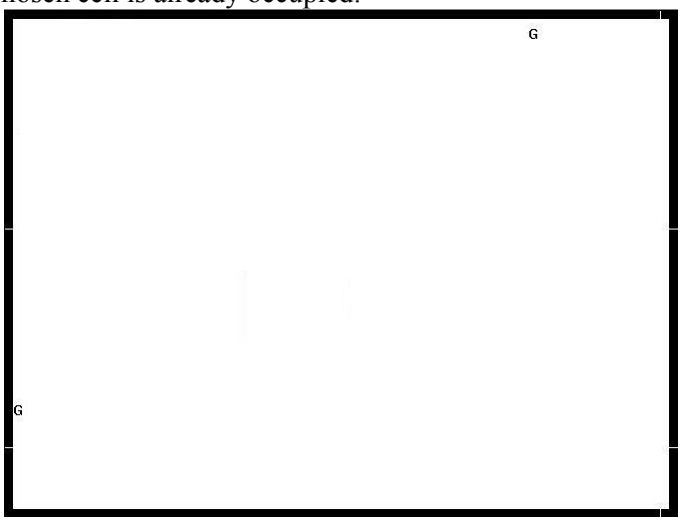


Figure 7. *A priori* map

Environment 7

The size of the environment is 2320×1710 mm. Real environment is represented in Figure 8. The environment is static and unknown *a priori* (shown in Figure 9).

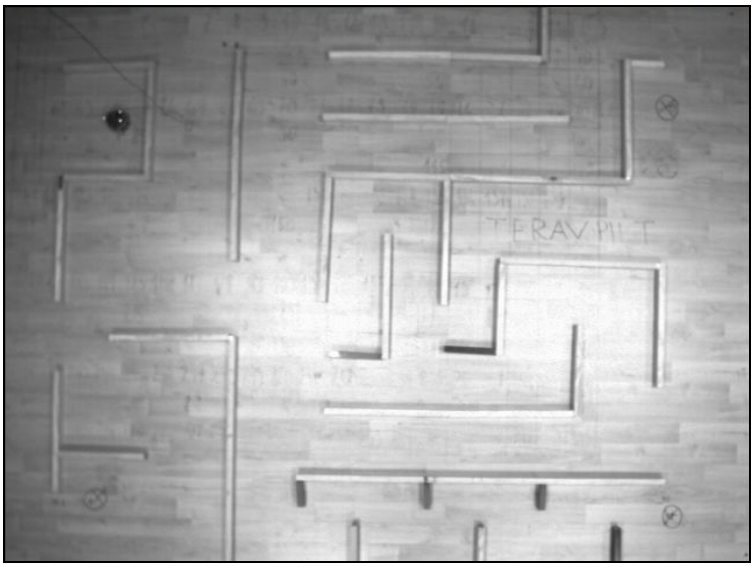


Figure 8. Real environment

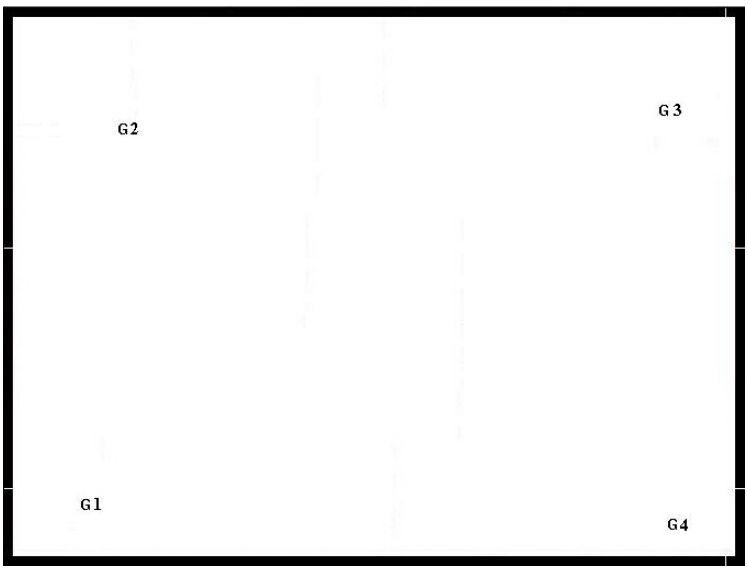


Figure 9. *A priori* map

APPENDIX F

Pseudo Code of the Learning Method

Introduction

This Appendix describes learning method used in appendices A, B, and C. The method is represented as pseudo code. The main function is `ExecutePlan()`. This function gets a predefined plan (goal points) and executes it.

Pseudo Code of the Learning Method

```
// Grid cell size (mm)
double GRID_SIZE = 29.2

// Similarity measure
double SIMILARITY_CONST = 2 * GRID_SIZE


// Main function
void ExecutePlan(Plan plan)
{
   while(plan.IsTaskLeft()) {
      DecisionMaker(plan);
   }
}


// Plan to the next goal point
void DecisionMaker(Plan plan)
{
   Cell start_cell;
   Cell goal_cell;
   GlobalMap map;

   start_cell = plan.GetNextStartCell();
   goal_cell  = plan.GetNextGoalCell();

   // Check for the old solution
   Case old_case = FindBestSimilarCase(Case(start_cell, goal_cell));
   if (AcceptOldSolution(old_case)) {   // chose old solution
      map.path = old_case.path;
   } else {                             //chose new solution
      map.path = GenerateSuboptimalPath(start_cell, goal_cell);
   }

   // Traverse to the next goal point using the path "map.path"
   map.ReactivePlanner();
   // The robot is arrived to the goal point

   double cost = CalcCost(map.replannings);
   Case new_case = Case(start_cell, goal_cell, map.relaxed_path, cost);
   AddToBase(new_case);
}
```

```
// Accept or do not accept old solution
bool AcceptOldSolution(Case old_case)
{
   if (old_case == NULL)
      return FALSE;
   if (random() < old_case.cost)
      return TRUE;
   else
      return FALSE;
}

// Find the best similar stored case (path) from the robot's base.
Case FindBestSimilarCase(Case current_case)
{
   Case best_similar_case = NULL;
   // check all stored cases in the base
   for (int i = 0; i < base.cases(); i++) {
      if (Max(Distance(current_case.start_cell, base[i].start_cell),
            Distance(curren_case.gaol_cell,  base[i].goal_cell)) <
            SIMILARITY_CONST) {
         if (best_similar_case == NULL) {
            best_similar_case = base[i];
         } else {
            if (best_similar_case.cost > base[i].cost) {
               best_similar_case = base[i];
            }
         }
      }
   }
   return most_similar_case;
}

// Calculate cost to the traversed path
double CalcCost(int replannings)
{
   int max_replannings = 30;
   if (replannings < 5)
      return 1.0;
   if (replannings > max_replannings)
      return 0;
   else
      return 1 - replannings / max_replannings;
}

// Store new case (path)
void AddToBase(Case new_case)
{
   Case most_similar_case = FindMostSimilarCase(new_case.path);
   if (most_similar_case != NULL) {
      if (new_case.cost < most_similar_case.cost) {
         base.DeleteAndAddCase(new_case);
      } else {
         most_similar_case.sum_cost += cost;
         most_similar_case.visits++;
         most_similar_case.cost = most_similar_case->sum_cost /
               most_similar_case->visits;
      }
   } else {
      base.AddCase(new_case);
   }
}
```

```
// Find most similar case (path)
Case FindMostSimilarCase(GlobalPath new_path)
{
   Case most_similar_case = NULL;
   double min_distance = MAX_NUMBER;

   // check all stored cases in the base
   for (int i = 0; i < base.cases(); i++) {
      distance = FindSimilarity(new_path, base[i].path);
      if (distance < min_distance) {
         min_distance = distance;
         most_similar_case = base[i];
      }
   }

   return most_similar_case;
}


// Find the similarity of the paths
double FindSimilarity(GlobalPath path1, GlobalPath path2)
{
   return MaxDistanceBetweenPaths(path1, path2);
}
```

APPENDIX G

CD - Website of the Experimental Data

Introduction

This Appendix is a website on the CD (also, available online at *http://math.ut.ee/~kristo/phd*) added to this thesis. On the website are available descriptions of the test environments, all experimental results (including the thumbnails of the traversed paths), and the program code. Open the file *index.html* on the CD for looking the website.

CURRICULUM VITAE

Kristo Heero

Citizenship:	Estonian Republic
Born:	February 21, 1977, Misso, Estonia
Marital status:	cohabit, 1 child
Address:	Pikk 12-5, 51009 Tartu, Estonia
Contacts:	e-mail: kristo@math.ut.ee

Education

1992 – 1995	Specialized math high school in Nõo
1995 – 2000	University of Tartu, Bachelor in Computer Science
2000 – 2002	University of Tartu, MSc in Computer Science

Professional employment

1998 – 2001	University of Tartu, system administrator
2001 – ...	Cybernetica AS, researcher and software engineer

Scientific work

The main fields of interest are path planning, robot learning and decision-making of the intelligent autonomous mobile robots.

Honours/Awards

- The stipend for doctoral student of information and communication technology of Tiger University, 2005
- The research stipend of the University of Munster, 2005
- The prize of Bernhard Schmidt of The Estonian Academy of Sciences for research and development of Estonian e-voting system, 2005

CURRICULUM VITAE

Kristo Heero

Kodakondsus:	Eesti
Sünniaeg ja –koht:	21. veebruar 1977, Misso, Eesti
Perekonnaseis:	vabaabielus, 1 laps
Aadress:	Pikk 12-5, 51009 Tartu
Kontaktandmed:	e-post: kristo@math.ut.ee

Haridus

1992 – 1995	Nõo Reaalgümnaasium
1995 – 2000	Tartu Ülikool, informaatika bakalaureus
2000 – 2002	Tartu Ülikool, informaatika magister

Erialane teenistuskäik

1998 – 2001	Tartu Ülikooli Arvutuskeskus, süsteemiadministraator
2001 – ...	Cybernetica AS, teadur, tarkvarainsener

Teadustegevus

Peamiseks tegevusvaldkonnaks on intelligentsete mobiilsete robotite tee-planeerimine ja õpistrateegiad.

Tunnustused

- Tiigriülikooli stipendium 2004 IKT doktorandile, 2004
- Münsteri Ülikooli uurimisstipendium, 2005
- Eesti Teaduste Akadeemia Bernhard Schmidti preemia teadus- ja arendustöö rakenduse eest: "e-hääletamise tarkvaralahendus", 2005

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

1. **Mati Heinloo.** The design of nonhomogeneous spherical vessels, cylindrical tubes and circular discs. Tartu, 1991, 23 p.
2. **Boris Komrakov.** Primitive actions and the Sophus Lie problem. Tartu, 1991, 14 p.
3. **Jaak Heinloo.** Phenomenological (continuum) theory of turbulence. Tartu, 1992, 47 p.
4. **Ants Tauts.** Infinite formulae in intuitionistic logic of higher order. Tartu, 1992, 15 p.
5. **Tarmo Soomere.** Kinetic theory of Rossby waves. Tartu, 1992, 32 p.
6. **Jüri Majak.** Optimization of plastic axisymmetric plates and shells in the case of Von Mises yield condition. Tartu, 1992, 32 p.
7. **Ants Aasma.** Matrix transformations of summability and absolute summability fields of matrix methods. Tartu, 1993, 32 p.
8. **Helle Hein.** Optimization of plastic axisymmetric plates and shells with piece-wise constant thickness. Tartu, 1993, 28 p.
9. **Toomas Kiho.** Study of optimality of iterated Lavrentiev method and its generalizations. Tartu, 1994, 23 p.
10. **Arne Kokk.** Joint spectral theory and extension of non-trivial multiplicative linear functionals. Tartu, 1995, 165 p.
11. **Toomas Lepikult.** Automated calculation of dynamically loaded rigidplastic structures. Tartu, 1995, 93 p. (in Russian).
12. **Sander Hannus.** Parametrical optimization of the plastic cylindrical shells by taking into account geometrical and physical nonlinearities. Tartu, 1995, 74 p.
13. **Sergei Tupailo.** Hilbert's epsilon-symbol in predicative subsystems of analysis. Tartu, 1996, 134 p.
14. **Enno Saks.** Analysis and optimization of elastic-plastic shafts in torsion. Tartu, 1996, 96 p.
15. **Valdis Laan.** Pullbacks and flatness properties of acts. Tartu, 1999, 90 p.
16. **Märt Põldvere.** Subspaces of Banach spaces having Phelps' uniqueness property. Tartu, 1999, 74 p.
17. **Jelena Ausekle.** Compactness of operators in Lorentz and Orlicz sequence spaces. Tartu, 1999, 72 p.
18. **Krista Fischer.** Structural mean models for analyzing the effect of compliance in clinical trials. Tartu, 1999, 124 p.
19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
20. **Jüri Lember.** Consistency of empirical k-centres. Tartu, 1999, 148 p.
21. **Ella Puman.** Optimization of plastic conical shells. Tartu, 2000, 102 p.

22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
25. **Maria Zeltser.** Investigation of double sequence spaces by soft and hard analitical methods. Tartu, 2001, 154 p.
26. **Ernst Tungel.** Optimization of plastic spherical shells. Tartu, 2001, 90 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 p.
28. **Rainis Haller.** *M(r,s)*-inequalities. Tartu, 2002, 78 p.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
30. **Eno Tõnisson.** Solving of expession manipulation exercises in computer algebra systems. Tartu, 2002, 92 p.
31. **Mart Abel.** Structure of Gelfand-Mazur algebras. Tartu, 2003. 94 p.
32. **Vladimir Kuchmei.** Affine completeness of some ockham algebras. Tartu, 2003. 100 p.
33. **Olga Dunajeva.** Asymptotic matrix methods in statistical inference problems. Tartu 2003. 78 p.
34. **Mare Tarang.** Stability of the spline collocation method for volterra integro-differential equations. Tartu 2004. 90 p.
35. **Tatjana Nahtman.** Permutation invariance and reparameterizations in linear models. Tartu 2004. 91 p.
36. **Märt Möls.** Linear mixed models with equivalent predictors. Tartu 2004. 70 p.
37. **Kristiina Hakk.** Approximation methods for weakly singular integral equations with discontinuous coefficients. Tartu 2004, 137 p.
38. **Meelis Käärik.** Fitting sets to probability distributions. Tartu 2005, 90 p.
39. **Inga Parts.** Piecewise polynomial collocation methods for solving weakly singular integro-differential equations. Tartu 2005, 140 p.
40. **Natalia Saealle.** Convergence and summability with speed of functional series. Tartu 2005, 91 p.
41. **Tanel Kaart.** The reliability of linear mixed models in genetic studies. Tartu 2006, 124 p.
42. **Kadre Torn.** Shear and bending response of inelastic structures to dynamic load. Tartu 2006, 142 p.
43. **Kristel Mikkor.** Uniform factorisation forcompact subsets of banach spaces of operators. Tartu 2006, 72 p.
44. **Darja Saveljeva.** Quadratic and cubic spline collocation for volterra integral equations. Tartu 2006, 117 p.

47.

48.

ISSN 1024–4212