Efficient Interpolated Path Planning of Mobile Robots based on Occupancy Grid Maps

Marija Đakulović * **Mijo Čikeš** * **Ivan Petrović** *

* *University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer (e-mail: {marija.dakulovic, mijo.cikes, ivan.petrovic}@fer.hr).*

Abstract: This paper focuses on interpolation-based path planning algorithms for mobile robot that use the occupancy grid map of the environment. Such an algorithm produces the path close to optimal solution in continuous search space, but has computational burden. We discuss on using heuristics to improve efficiency of the E* algorithm, as the interpolation-based path planning algorithm, with a trade-off of generating less optimal paths. We compare obtained paths with the almost optimal solution in continuous search space and also with the path obtained by the D* algorithm, which is a representative of the classical graph search algorithms applied on the occupancy grid map. All used algorithms are verified both by simulation and experimentally on a Pioneer 3DX mobile robot equipped with a laser range finder.

Keywords: path planning, graph search, autonomous mobile robots.

1. INTRODUCTION

This paper considers the problem of path planning for autonomous mobile robots in dynamic environments. An important issue for the path planning algorithm is the shape of obtained path, e.g. the number of points in which path changes directions, since in these points the robot must slow down or even stop and turn in place. Furthermore, the algorithm real-time performance is very important in changing environments. It means that the path must be replanned as the environment changes to prevent collisions with the newly detected obstacles.

The majority of path planning algorithms for mobile robots use equally-spaced grid of discrete points (cells), called the occupancy grid map, to represent a continuous environment (Thrun et al. (2005)). From such a grid representation a graph can be created, where nodes are grid cells, edges are connections between neighbor grid cells, and weights are costs of traversing edges. The path planning algorithm is then considered as a graph search algorithm (Russell et al. (1995)). The main representative is the A* algorithm, which uses the heuristic function to focus the search on the minimal number of nodes necessary to find the optimal path (Hart et al. (1968)). Stentz (1994) developed the D* algorithm as a dynamic version of the A* algorithm, which is more efficient in environments with changing or unknown obstacle configuration.

The path obtained by a graph search algorithm is composed of neighboring nodes in the graph (grid points), and therefore, is constrained to eight directions of path segments. On the contrary, the interpolation-based path planning algorithms produce paths whose points are interpolated between grid points, e.g. the Field D* algorithm (Ferguson and Stentz (2007)) or the E* algorithm (Philippsen and Siegwart (2005)). The Field D* algorithm uses linear interpolation to derive path points between grid intersections. Similarly, the E* algorithm uses interpolation to produce the approximation of the continuous optimal path costs at grid points. The path is obtained by following the gradient descent of calculated path costs. In our previous work (Đakulović and Petrović (2011)) we proposed the algorithm called Two-Way D* (TWD*), which calculates the shortest possible path in the geometrical space by connecting certain non-neighbor grid points. Above mentioned algorithms produce more natural low-cost paths through grids with continuous headings, but have about twofold higher computational time than the basic D* algorithm. However, their efficiency can be increased by using heuristics to focus the search. Philippsen (2006) states that heuristics together with interpolation cause different and more complicated behavior of the algorithm, which has not been resolved yet for the E* algorithm. Namely, the optimal interpolated cost is not guaranteed to be found. For the Field D*, Ferguson and Stentz (2005) state that the standard Euclidean cost dividing by two has shown itself as the best heuristics in practice as it is more efficient and produces solutions that are not noticeably different from the optimal solutions. Ferguson et al. (2005) present comprehensive study of heuristic-based search algorithms without interpolation, but review on incorporating the heuristics into the interpolation-based path planning algorithms has not been done yet.

Recently, we present comparation of D*, E* and TWD* algorithms (Čikeš et al. (2011)). Here we extend our previous comparative study by examining heuristics influence on

⋆ This research has been supported by the Ministry of Science, Education and Sports of the Republic of Croatia under grant No. 036 − 0363078 − 3018 and by the European Community's Seventh Framework Programme under grant No. 285939 (ACROSS).

the E* algorithm. Heuristics is scaled by a varying factor in order to inspect dependency on path quality. Obtained paths are compared to the path of the TWD* and D* algorithm according to certain path characteristics. Additionally, parameters related to the real-time performance of all three algorithms are also compared.

The rest of the paper is organized as follows. The problem formulation is given in section 2, section 3 presents compared algorithms in short, section 4 presents simulations results and experimental results are given in section 5.

2. THE OCCUPANCY GRID MAP AND SEARCH GRAPH

An occupancy grid map is created by approximate cell decomposition of the environment (Thrun et al. (2005)). The whole environment is divided into squared cells of equal size e_{cell}, which are abstractly represented as the set of M elements $\mathcal{M} = \{1, \ldots, M\}$ with corresponding Cartesian coordinates of cell centers $c_i \in \mathbb{R}^2$, $i \in \mathcal{M}$. Each cell contains occupancy information $o : \mathcal{M} \mapsto \mathbb{R}$ of the part of the environment that it covers.

In this paper two types of occupancy grid maps are used: binary occupancy grid maps and weighted occupancy grid maps. In a binary occupancy grid map occupancy function o contains only two values, 1 for free and ∞ for occupied cells. In a weighted occupancy grid map additional occupancy values are used for cells that are close to obstacles. We use simple procedure of calculating occupancy values:

$$o(i) = \begin{cases} \max\{1, (M_c + 2 - \min\limits_{j \in \mathcal{O}} \|c_i - c_j\|_\infty)\} & \text{if } i \notin \mathcal{O} \\ \infty \quad \text{if } i \in \mathcal{O} \end{cases} \tag{1}$$

where $\|\cdot\|_\infty$ is the infinity norm, M_c is the integer number of cells defining the size of the safety cost mask around obstacles, and $\mathcal{O}$ is the set of obstacle cells. It is assumed that all obstacles are enlarged to account for robot's dimensions, where the robot can be approximated by a circle.

Weighted undirected search graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ is created from the occupancy grid map in such a way that all unoccupied cells $\mathcal{N} = \mathcal{M} \setminus \mathcal{O}$ represent the set of nodes in the graph. Edges are defined between neighbor nodes $\mathcal{E} = \{\{i, j\} \mid i, j \in \mathcal{N}, i \text{ and } j \text{ are neighbors}\}$, where two nodes $i, j \in \mathcal{N}$ are neighbors if $\|c_i - c_j\|_\infty = e_{cell}$. Weights are the cost of transition between neighbors $\mathcal{W} = \{w_{i,j} \mid i, j \in \mathcal{N}, i \text{ and } j \text{ are neighbors}\}$:

$$w_{i,j} := \|c_i - c_j\| \cdot \max\{o(i), o(j)\}. \tag{2}$$

In binary occupancy grid maps there are two values of transitions between neighbors: straight and diagonal transition. In weighted occupancy grid maps transitions between neighbors are additionally weighted according to their distances to the obstacles.

A path $\mathcal{P}$ in the search graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ is defined as a succession of neighboring nodes, i.e. the points in the path are neighboring cell centers (e.g. the path calculated by the D* algorithm). For the interpolation-based path planning algorithm a path $\mathcal{P}_{int}$ is composed of points that are located arbitrary within cells (e.g. the path calculated by the E* algorithm). The cost of the path in the search graph $\mathcal{P} = (n_1, n_2, \ldots, n_L)$ is defined as the sum of weights of edges along the path, i.e.,

$$c(\mathcal{P}) := \sum_{i=1}^{L-1} w_{n_i, n_{i+1}}. \tag{3}$$

The cost of the path $\mathcal{P}_{int} = (c_1, c_2, \ldots, c_{L_{int}})$ is computed similarly:

$$c(\mathcal{P}_{int}) := \sum_{i=1}^{L_{int}-1} \|c_i - c_{i+1}\| \cdot \max\{o(n_i), o(n_{i+1})\}, \tag{4}$$

where nodes n_i and n_{i+1} are determined as cells that contain coordinates c_i and c_{i+1}, respectively. Here is assumed that two consecutive path coordinates are close enough so they belong to the same cell or to two neighbor cells. If c_i and c_{i+1} belong to distanced non-neighbor cells then equally spaced points lying on the line segment $\overline{c_i c_{i+1}}$ need to be included for the path cost calculation.

Figure 1 represents the weighed occupancy grid map with $M_c = 4$ cells wide safety cost mask of the section of the test environment. Free space is colored white ($o(\cdot) = 1$), unoccupied space within the safety cost mask is colored by shades of gray ($o(\cdot) \in \{2, 3, 4, 5\}$), and obstacles are colored black ($o(\cdot) = \infty$), where real obstacle positions are marked by x. For illustration, two paths are shown, the path calculated by the D* algorithm and the path calculated by the E* algorithm. Notice that the number of cells M_c influence on the distance between the path and the obstacles.

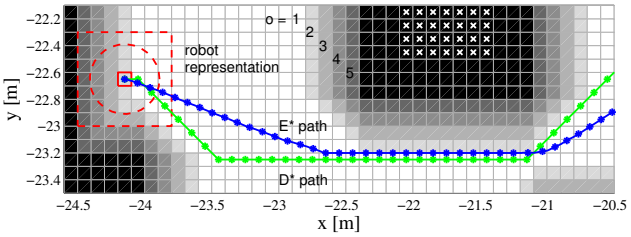

Fig. 1. A section of the test environment represented by the weighted occupancy grid map with the safety cost mask ($M_c = 4$).

3. PATH PLANNING

This section briefly restates the algorithms D*, TWD* and E*. More details can be found in Stentz (1994), Philippsen and Siegwart (2005) and Đakulović and Petrović (2011), respectively.

3.1 The D algorithm*

For every searched node n, the D* algorithm computes the cost value $g(n)$ of the optimal path from the node n to the goal node. The algorithm stores the *backpointers* for every searched node n, which point to the parent node with the smallest cost g. A backpointer is noted by the function $b(\cdot)$, where $b(n) = m$ means that the node n has the smallest cost because it follows the node m. The backpointers ensure that optimal path from any searched node n to the goal node can be extracted according to the function $b(\cdot)$.

The D* algorithm starts the search from the goal node, examining neighbor nodes of minimal g value until the start node is reached. For better replanning performances, the exhaustive search can be done in the initial phase, which computes optimal paths and path costs g from every reachable node in the graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ to the goal node.

If during the motion robot detects change of occupancy values, first the weights in the graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ are updated according to (2). The backpointers are redirected locally and the new optimal path from the robot's current position is determined. The number of expanded nodes is minimal and consequently the time of execution. A simple illustration of the algorithm can be seen in Fig. 2.

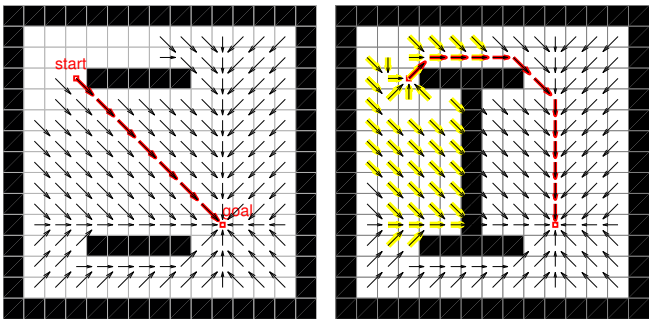

Fig. 2. Illustration of path planning (left) and replanning (right) by the D* algorithm.

3.2 The TWD algorithm*

The TWD* algorithm does the exhaustive search of the graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ by applying one pass of the D* algorithm executed from the goal node to the start node (standard D*) and another pass executed from the start node to the goal node (the reverse D* (RD*)). The sum of the path costs of both D* passes, costs g for D* pass and g_R for RD* pass give the complete cost of the path $f(n) = g(n) + g_R(n)$.

The set of all nodes n such that $f(n) = f_{min}$ forms the geometrical area of minimal cost f_{min}. The path is calculated by partitioning the minimal cost area into smaller convex polygons and by finding the connection of polygon vertices that form the shortest path in geometrical space. The shortest path found by the TWD* algorithm consists of straight line segments with each straight line segment lying within the area of minimal cost f_{min} (see Fig. 3).

3.3 The E algorithm*

The E* algorithm uses the interpolation function, which assigns numerical value to each cell by the so called wavefront propagation over the free cells in the grid map. The wavefront propagation acts like a continuous contour that sweeps from the goal node outwards and at each cell record the crossing time. The path cost at each cell can be calculated by dividing the crossing time by the speed of the propagation within the cell $F(n) = 1 - r(n)$, where $r(n)$ is the risk of traversal through the cell n. The risk function r is normalized to $[0, 1]$ where risk 0 corresponds to free cells, and risk 1 corresponds to occupied cells. The lower r

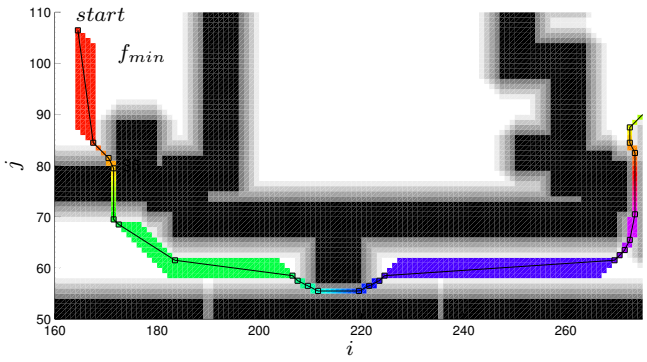

Fig. 3. Illustration of the path calculation by the TWD* algorithm.

implies higher propagation speed. The occupancy function o given by (1) can be transformed to the risk function r by normalization

$$r(i) = \min\left(1, \frac{o(i) - 1}{M_c + 1}\right) \tag{5}$$

Therefore, for the example in Fig. 1 occupancy values $o(i) \in \{1, 2, 3, 4, 5, \infty\}$ can be transformed to risk values $r(i) \in \{0, 0.2, 0, 4, 0.6, 0.8, 1\}$.

Unlike the D* algorithm, which uses one backpointer for a node n to represent which node is the next node in the optimal path ($b(n) = m$), or on which neighbor node m the path cost $g(n)$ depends, the E* algorithm uses backpointers for one or two neighbors that are not obstacles and are not lying on same axes. For the E* algorithm, each node has four neighbors in undirected graph, diagonal cells are omitted. Each node n has two values – $rhs(n)$ and $v(n)$. Nodes that wait to be expanded are sorted in the queue called *Wavefront* by the key $\min(rhs(n), v(n))$. When the node n is subtracted from the queue the value $v(n)$ become equal to $rhs(n)$. The path is obtained by following the gradient descent of calculated path costs. A simple illustration of the algorithm can be seen in Fig. 4.

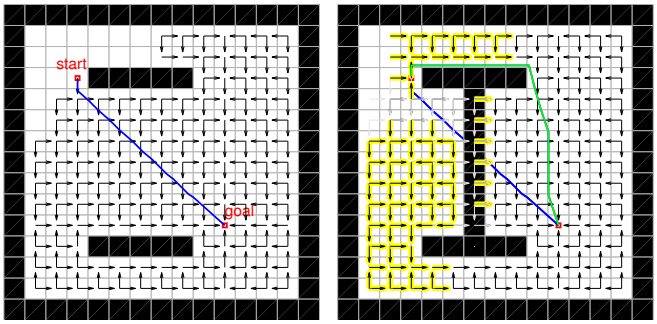

Fig. 4. Illustration of path planning (left) and replanning (right) by the E* algorithm.

Since the E* is based on the D* algorithm which is capable of focused heuristic search to speed up planning, heuristics can be incorporated in the key for sorting the queue Wavefront as $\min(rhs(n) + k \cdot h(n), v(n))$, where h is heuristic function and k is a scalar factor. The most common used heuristic function is Euclidean distance to the goal, which is used in this work.

4. SIMULATION RESULTS

We tested heuristics influence on the E* algorithm by changing the factor $k \in \{0, 0.25, 0.5, 1\}$ and comparing obtained paths and execution parameters to the D* and TWD* algorithms. Tests were performed on three different occupancy grid maps. The first map is the free space, i.e. the environment without obstacles shown in Fig. 5, the second map is the Department map, which is a real experimental environment of our Department shown in Fig. 7 and the third map is randomly generated map shown in Fig. 9. The first two maps are represented by a binary occupancy grid map, while the third one is represented by a weighted occupancy grid map. Simulation results are measured on notebook with Intel Core2Duo T7500 processor on Ubuntu 10.04 Linux operating system. Measurements were done in optimization mode (opt) with -03 flag on GNU C compiler. Path characteristics are: l - length of the initial path; c - cost of the initial path, which is the same as the length for calculations in binary occupancy grid maps; $n\alpha_{init}$ - the number of points in which the initial path changes direction (changes less than 5° were ignored); $\Sigma\alpha_{init}$ - the sum of all angles of the initial path direction changes. The execution parameters for the initial planning are the number of algorithm iterations noted by I and corresponding time noted by t_{init}. On the Department map we also tested replanning in case of changes in the environment.

4.1 Free space

The dimension of the map is 165×540 cells, $e_{cell} = 0.1$ m. Numerical results are given in Table 1 and the path comparison is shown in Fig. 5 and 6. The TWD* path is the shortest straight line path ($n\alpha_{init} = \Sigma\alpha_{init} = 0$), and both E* without heuristics and TWD* algorithm give significantly shorter path then the basic D* algorithm for about 7%, but the time of execution t_{init} is worse for a factor 1.5. Shorthand exh. in Table 1 means that exhaustive initial search is performed. Executional time is lowered by using heuristics in the E* algorithm. For all values of k initial search is performed regularly, i.e., until sufficient nodes are searched to find the path. For $k = 0.5$ executional time is lowered for about 24%. The largest decrease of executional time is for $k = 1$ for about 80%, but the path differs significantly from the basic E* path, as can be seen in Fig. 6, although it is still shorter than the D* path. The E* for $k = 1$ has the largest values of $n\alpha_{init}$ and $\Sigma\alpha_{init}$.

Table 1. Comparison of algorithms in the free space

Algorithm	I	t_{init} [ms]	$n\alpha_{init}$	$\Sigma\alpha_{init}$ [°]	l [m]
D*	82150	338	1	45	28.458
TWD*	164300	484	0	0	26.514
E*(exh.)	82150	486	3	31	26.526
E*(k=0)	81069	486	3	31	26.526
E*(k=0.25)	61629	448	3	33	26.527
E*(k=0.5)	46776	370	4	46	26.533
E*(k=1)	10119	86	79	189	27.038

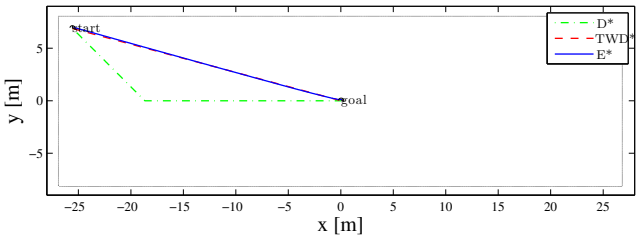

Fig. 5. Comparison of paths in the free space.

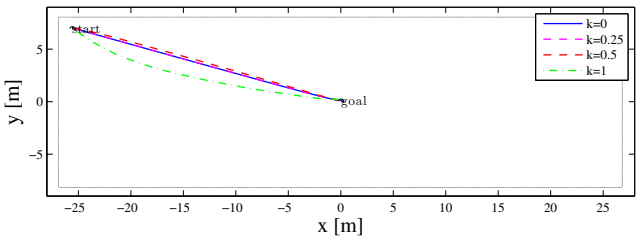

Fig. 6. Comparison of paths of the E* algorithm for heuristics factors $k \in \{0, 0.25, 0.5, 1\}$ in the free space.

4.2 Department map

The dimension of the map is 165×540 cells, $e_{cell} = 0.1$ m. Numerical results for initial planning are shown in Table 2 and the path comparison is shown in Fig. 7 and 8. The results are similar to the ones obtained on the empty map. The TWD* algorithm produces the shortest path, the path is shorter than the basic D* algorithm for about 6 %. The E* algorithm without heuristics has shorter path for about 4 % than the basic D* algorithm. The time of execution t_{init} for the E* with exhaustive search and TWD* algorithms are worse than the basic D* algorithm for about 4 times. By using heuristics in the E* algorithm time of execution can be lowered, as can be seen in Table 2. By using heuristics with a factor $k = 1$ time of execution is lowered to the time of the D* algorithm, but the path grows longer and is shorter than the path obtained by the basic D* algorithm for about 0.6 %. However, even for small factor value $k = 0.25$ the path is very similar to the path obtained by the E* algorithm without heuristics and execution time is lowered significantly by a factor 2.

Table 2. Comparison of algorithms in the Department map

Algorithm	I	t_{init} [ms]	$n\alpha_{init}$	$\Sigma\alpha_{init}$ [°]	l [m]
D*	32724	26	15	675	18.81
TWD*	65448	102	11	239	17.65
E*(exh.)	32724	119	20	490	18.03
E*(k=0)	18517	79	20	490	18.03
E*(k=0.25)	14791	63	21	500	18.04
E*(k=0.5)	11736	49	28	719	18.16
E*(k=1)	3860	16	79	848	18.69

4.3 Randomly generated map

The dimension of the randomly generated map is 500×500 cells, with the cell size of $e_{cell} = 0.1$ m and of about 50% occupied cells. Obstacles were created by drawing the squares at random places and of random sizes. Numerical

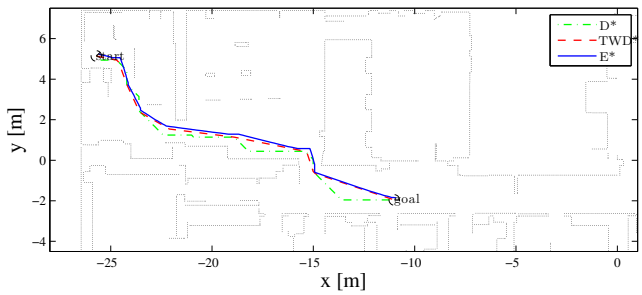

Fig. 7. Initial paths in the Department map.

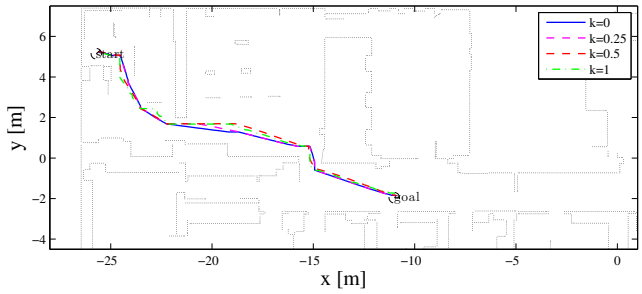

Fig. 8. Initial paths of the E* algorithm for heuristics factors $k \in \{0, 0.25, 0.5, 1\}$ in the Department map.

results of the algorithms comparison is given in Table 3, and the path comparison is shown in Fig. 9 and 10. Since this map is represented by the weighted occupancy grid map, value l (the length of the initial path) and c (the cost of the initial path) differ. Expectingly, the TWD* algorithm has the smallest cost of the path, but E* is very close to TWD*. The both E* and TWD* algorithm give significantly shorter path then the basic D* algorithm for approximately 6 %, but the time of execution t_{init} is worse for a factor 2. By using heuristics executional time is lowered for $k = 1$ below the executional time of the D* algorithm, but path parameters $n\alpha_{init}$ and $\Sigma\alpha_{init}$ grow to high numbers.

Table 3. Comparison of algorithms in the randomly generated map

Algorithm	I	t_{init} [ms]	$n\alpha_{init}$	$\Sigma\alpha_{init}$ [°]	l [m]	c [m]
D*	161609	979	46	2070	62.11	63.45
TWD*	323218	2124	27	600	59.44	60.51
E*(exh.)	161609	1575	121	1498	58.75	60.95
E*(k=0)	160761	1551	121	1498	58.75	60.95
E*(k=0.25)	148081	1624	124	1494	58.82	61.02
E*(k=0.5)	122141	1429	160	1920	59.12	61.34
E*(k=1)	56925	561	458	12332	61.21	64.54

4.4 Path replanning

Replanning capability was tested on the Department map. Simulations were repeated 10 times for each value of the parameter k and results were averaged. At each simulation run artificial obstacles were moving through the environment and blocking passages, as can be seen in Fig. 11. Table 4 shows times of replanning on the Department map. t_{rep}^{max} denotes the highest time of replanning during the robot motion through the environment populated

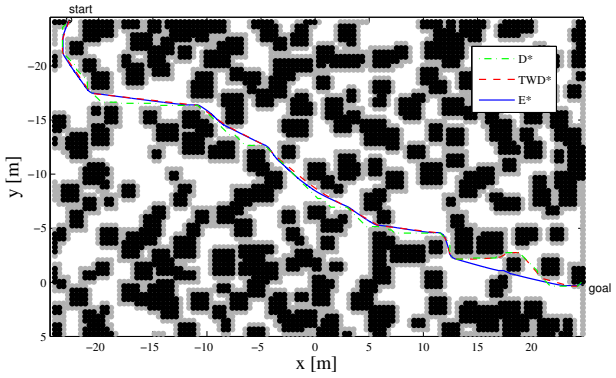

Fig. 9. Path comparison in the randomly generated map with the safety cost mask.

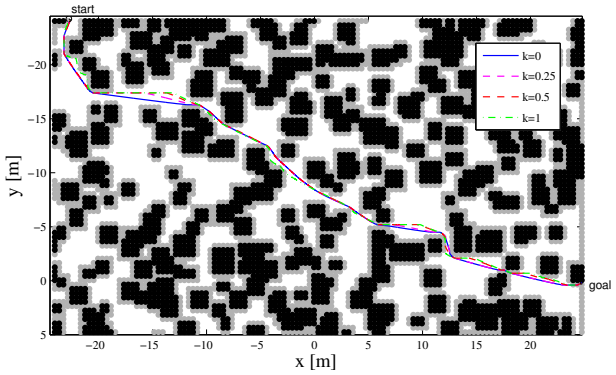

Fig. 10. Path comparison of the E* algorithm for heuristics factors $k \in \{0, 0.25, 0.5, 1\}$ in the randomly generated map with the safety cost mask.

by unknown obstacles. The number of iterations of the highest replanning is denoted by I_{rep}^{max}. The sum of all replanning times is denoted by Σt_{rep} and ΣI_{rep} is the sum of all iterations during replannings. It can be noticed that replanning execution times depend on whether the initial search is performed exhaustively (shorthand exh. in Table 4), i.e. all nodes are searched regardless the path was found some iterations ago, or regularly, i.e. only sufficient nodes are searched to find the path. If regular search is performed, it can happened that algorithm needs more time for replanning if some new nodes, which were not searched initially must be examined. E* with exhaustive initial search and TWD* have similar times of replanning, but the basic D* algorithm has the best replanning time and t_{rep}^{max} is better approximately for a factor 2. However, the replanning times of the heuristic E* algorithm with the exhaustive initial search and factor $k = 1$ is very close to the replanning times of the D* algorithm. Even for small factor value $k = 0.25$ replanning times are lowered.

5. EXPERIMENTAL RESULTS AND CONCLUSIONS

The experimental results were obtained with a Pioneer 3DX mobile robot at our Department. The occupancy grid map with the safety cost mask was used. The laser range finder SICK LMS200 mounted on the robot was used for environment perception. The dynamic window based algorithm, described in our previous work (Seder et al. (2005)), was used for path following. Two experiments

Table 4. Replanning execution

Alg.	t_{rep}^{max} [ms]	I_{rep}^{max}	Σt_{rep} [ms]	ΣI_{rep}
D*	11	9567	38	18051
TWD*	20	18885	139	75344
E*(exh.)	22	6676	115	30943
E*(k=0)	64	18684	187	61101
E*(exh. k=0.25)	17	5905	103	18821
E*(k=0.25)	21	6945	109	19006
E*(exh. k=0.5)	16	5801	112	19378
E*(k=0.5)	28	7417	127	24319
E*(exh. k=1)	15	4619	131	31132
E*(k=1)	29	8997	146	39469

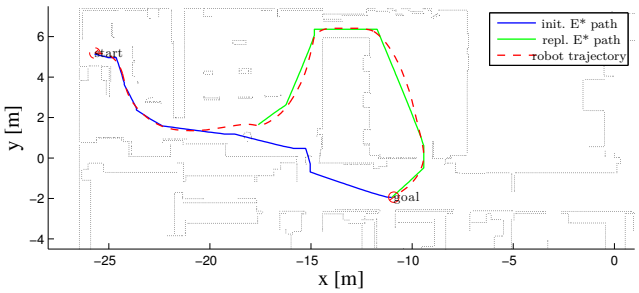

Fig. 11. Initial and replanned paths by the E* algorithm in the Department map.

were made: the first without heuristics ($k = 0$) and the second with heuristics multiplied by a factor $k = 0.25$ since this factor has shown to produce very similar paths to the paths obtained by the original E* algorithm and also to lower the computational time.

Fig. 12 shows the initial and replanned paths calculated by the E* algorithm, and the robot's trajectory while following the path. Laser readings are shown from all robot's positions. According to the laser readings it can be noticed that the door was closing while the robot tried to pass through it.

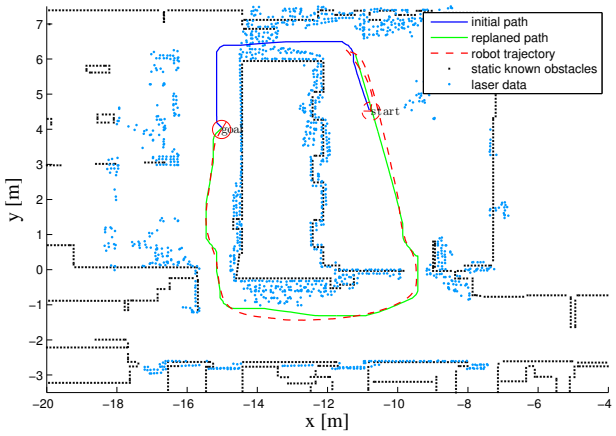

Fig. 12. The initial and replanned paths calculated by the E* algorithm and the trajectory driven by the mobile robot Pioneer 3DX.

Numerical results of the replanning is shown in Table 5. It can be seen that the sum of all replannings is much higher comparing to the simulation results, which is due to incomplete map of the environment. It can also be noticed that using heuristics multiplied by a factor 0.25 lowered number of explored nodes and computational times.

Table 5. Replanning execution

Alg.	t_{rep}^{max} [ms]	I_{rep}^{max}	Σt_{rep} [ms]	ΣI_{rep}
D*	17	5286	161	67457
TWD*	23	9618	419	100579
E*(k=0)	14	4197	1217	135190
E*(k=0.25)	14	3842	1072	105600

REFERENCES

Ferguson, D., Likhachev, M., and Stentz, A. (2005). A guide to heuristic-based path planning. In *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, 9–18.

Ferguson, D. and Stentz, A. (2005). The field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-05-19*.

Ferguson, D. and Stentz, A. (2007). Field D*: An Interpolation-Based Path Planner and Replanner. *Robotics Research: Results of the 12 th International Symposium ISRR (STAR: Springer Tracts in Advanced Robotics)*, 28, 239–253.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Philippsen, R. (2006). A light formulation of the E* interpolated path replanner. *Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, Tech. Rep.*

Philippsen, R. and Siegwart, R. (2005). An interpolated dynamic navigation function. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, volume 4, 3782–3789. Citeseer.

Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (1995). *Artificial intelligence: a modern approach*. Prentice Hall Englewood Cliffs, NJ.

Seder, M., Maček, K., and Petrović, I. (2005). An integrated approach to real-time mobile robot control in partially known indoor environments. *Industrial Electronics Society, 2005. IECON 2005. 32nd Annual Conference of IEEE*, 1785–1790.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 3310–3317.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. Cambridge, Massachusetts: MIT Press.

Čikeš, M., Đakulović, M., and Petrović, I. (2011). The path planning algorithms for a mobile robot based on the occupancy grid map of the environment – a comparative study. In *Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on*, 1–8. IEEE.

Đakulović, M. and Petrović, I. (2011). Two-way D* algorithm for path planning and replanning. *Robotics and Autonomous Systems*, 59(5), 329–342.