A COMBINED APPROACH OF PATH PLANNING FOR MOVING OBJECTS IN VIRTUAL ENVIRONMENTS

Gašper Mušič[1], **Gregor Klančar**[1]

[1]University of Ljubljana, Faculty of Electrical Engineering
1000 Ljubljana, Tržaška 25, Slovenia

gasper.music@fe.uni-lj.si (Gašper Mušič)

Abstract

The paper compares selected methods of path planning with regard to the computational complexity with the goal of establishing a framework for path planing of moving objects in virtual environment. The path planning framework is intended for use in various training simulators, e.g. in cases where the trainee is driving a simulated vehicle while there are other autonomously moving vehicles within the virtual environment. The trajectories of autonomously moving objects have to be carefully planned in order to obtain a realistic performance of a simulator. Based on the results of the comparison of the path planning methods, the advantages and drawbacks of selected methods are identified and a new combination is proposed with a suboptimal but efficient corridor calculation and an advanced path optimization within the corridor. The combined approach utilizes heuristic search algorithms as well as methods of numerical solving of a particular form of partial differential equation - an eikonal equation. The use of related fast marching method enables to derive smooth trajectories within the corridor identified by the heuristic search algorithm while keeping the on-line computational burden relatively low.

Keywords: virtual environments, path planning, quadtrees, triangulation, fast marching method.

Presenting Author's Biography

Gašper Mušič received B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Ljubljana, Slovenia in 1992, 1995, and 1998, respectively. He is Associate Professor at the Faculty of Electrical Engineering, University of Ljubljana. His research interests are in discrete event and hybrid dynamical systems, supervisory control, production control and industrial informatics.

1 Introduction

Virtual environments, such as training simulators, contain a number of moving objects. While some of them are manually driven by the operator, others move autonomously to simulate a dynamic environment. In order to obtain a realistic performance of a simulator, the moving paths have to be carefully planned according to object target points and current environment configuration. In dynamic, changing virtual environment the path calculations for an autonomous moving object need to be frequently updated, therefore making efficient set of path planning algorithms one of the key components.

With the given map of the environment and the target location the path planning aims to determine a trajectory, which will lead the object from the starting position to the target position. In general the planning involves two stages: (i) a suitable representation of the environment where the path has to be planned, and (ii) a search algorithm, that is capable of finding (sub)optimal path from the initial to the target position. The planning can involve a third stage where the path is optimized taking into account dynamic constraints of a moving object. This often leads to a requirement for smooth moving paths free of sharp turns.

The path planning methods initially transform the environment where the object of interest resides into a structure adapted to the requirements of path planning. Such representations include generalized Voronoi diagrams, various methods of space triangulation, regular grids and quadtrees, among others. Particular path planning algorithms may be used in combination with different representations of the environment, although certain representations are more suitable for some algorithms. Most generally used algorithms include A* and its derivates D*, D* Lite, E*, algorithms based on fast marching method and others.

The computational burden of A* algorithm (and others) increases rapidly with the increase of the search space. A better efficiency can be achieved by two level planning where a coarse plan is derived first and fine planning takes place within constrains imposed by the initial plan.

The literature review indicates environment segmentations based on Delaunay triangulation (dual to Voronoi diagram) and framed quadtrees among the most promising approaches, and A* based path search algorithms as a suitable way of path optimization within the segmented environment. The computational demand of planning in complex environments is reduced by two or three level planning and exploitation of a-priori knowledge or heuristics. Alternative approaches, e.g. bug algorithms, are also investigated but their application is questionable as the results are often not predictable, calculated path may be far from optimal or the target is not reached.

The obstacle avoidance for the known fixed obstacles is included in the path planning algorithms. The moving obstacles need to be addressed separately. Avoidance of moving obstacles can be achieved by efficient suboptimal approaches such as potential (vector) field which repels objects from obstacles and attract them to the target or the previously planned path. The avoidance is achieved here in a purely responsive manner, without real planning. The planning based approaches often involve D* algorithm, i.e. a dynamic version of the A* algorithm, where the branch costs within the calculated search tree may be readjusted during optimization when a change in environment occurs, and the complete recalculation of the tree is not necessary as in case of A* algorithm.

In the presented work a simulation study of a selected number of path planning algorithms has been performed with the emphasis on their computational demand. Based on the results, a novel combination of path planning algorithms has been proposed with a suboptimal but efficient corridor calculation and an advanced path optimization within the corridor.

2 Search space segmentation based path planning methods

Two space segmentation methods were considered in combination with an A* star based path optimizer: Quadtree segmentation and Delaunay triangulation.

2.1 Quadtrees space segmentation

Quadtrees enable a decomposition of the space map into free cells and cells occupied by obstacles. Cells are quadratic with the varying dimension. The map is initially divided by a coarse rectangular grid of four cells. The cells of the grid which are partially covered by obstacles are further divided until a prescribed description accuracy is achieved.

The last level of cells without children (successors) are called leafs which can be occupied with some obstacle or free. An essential step in quadtree generation is the cell occupancy test which is due to computational efficiency performed in two steps:

1. If squares outlined to the obstacles have no intersection with currently tested cell then this cell is marked as free leaf otherwise proceed to step 2.

2. Check if any of the obstacle corner is inside of the tested cell or if any of the cell corners is inside of the obstacle. If none of the two is true, additionally check if any of the cell and the obstacle sides cross. If none of the tests is true than mark this cell as a free leaf otherwise divide the cell and proceed with step 1.

Quadtree segmentation results in compact environment map presentation and enables efficient query about occupancy of some position or area in the environment [7] [8].

To shorten the computational time of A* path finding algorithm the obtained quadtree is extended with visibility graph which indicates possible paths among free cells. To each free leaf cell an array of its free neighbor

indexes is adjoined. This enables easier and computationally efficient moves of A* algorithm among the free quadtree leafs. The visibility graph is computed as follows:

- query for the area that is a little bigger ($1/2$ size of the prescribed minimal cell size) than current leaf size
- find leafs that are visible (accessible) from current leafs. Store their indexes to the current leaf visibility array,
- Calculate distances to the visible neighbor leafs and store them in an array.

By this stage the obtained quadtree structure is prepared for A* path finding algorithm. The pathfinding algorithm based on A* search strategy guarantees the shortest path between start and end point if an optimistic heuristic is used. This means that predicted path cost (length) must always be smaller (shorter) or at lest equal to the real path cost (length). Choosing line of sight for the predicted path length always fulfils this condition.

A corresponding example of the space segmentation and a calculated sample path are shown in Figure 1.

2.2 Space segmentation by constrained Delaunay triangulation

The triangulation is an important tool for representation of planar regions and is used in several application. The planar region is divided into a number of sub-regions of triangular shape. Commonly a set of points in the plane is given and the vertices of sub-regions must match the given set. This can be achieved in several ways, one of the possible triangulations is the Delaunay triangulation (DT).

The Delaunay triangulation assures a minimal number of narrow triangles (with small internal angles) which is a required property in many applications. The algorithms of Delaunay triangulation are also well explored and efficient algorithms yield computational complexity of $n \log n$ where n is the number of given points.

When certain edges are fixed in advance and the rest of the edges are determined in the DT way, the triangulation is called Constrained Delaunay Triangulation (CDT). The pre-defined edges are constraints during triangulation. CDT is useful in case where there are objects in the plane which should be taken into account during plane segmentation. This is the case when path should be planned in a plane with obstacles.

The triangulated plane can be used for path planning in a similar manner as the Quadtree based segmentation. A connectivity graph is built which contains information about the edges that can be crossed, i.e. the edges that are do not belong to obstacles. Based on the triangulation and the connectivity graph, a path can be searched for by an A* type algorithm. A corresponding example for the same environment configuration as in Figure 1 is shown in Figure 2. The additional thin edges drawn within the triangles show the alternative paths explored by the algorithm. It can be observed that the built-in heuristic helps the algorithm to search only a part of the whole space.

2.3 Path smoothing and shortening

As can be observed from in Figures 1 and 2 the obtained path passes through centers of the cells in quadtree space representation or crosses midpoints of the triangle edges in triangulation. None of the two approaches gives a smooth or straight path. The path is cornered even in the areas with no obstacles, which is not desired from the practical point of view, since the movement of

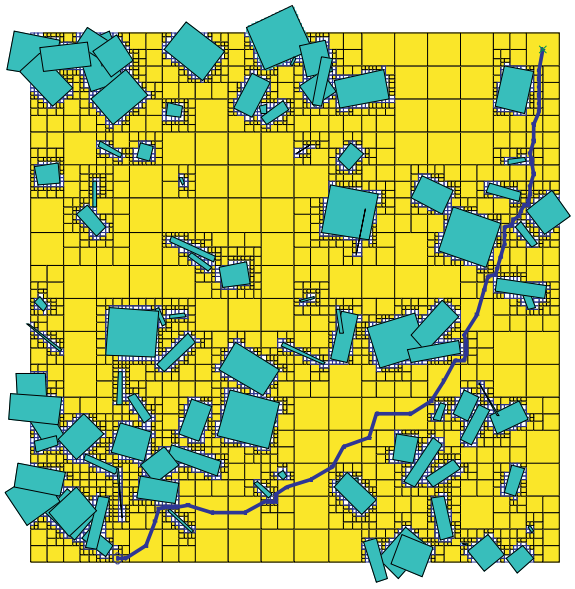

Fig. 1 Example of path planning based on Quadtree segmentation

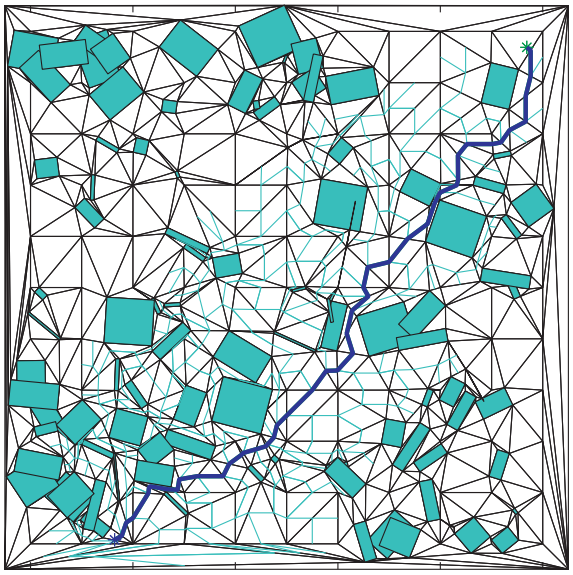

Fig. 2 Example of path planning based on triangular segmentation

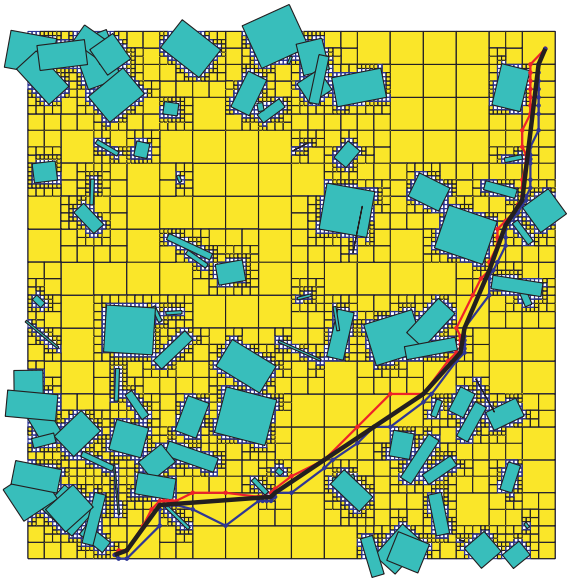

Fig. 3 Example of the optimal path (thick dark line) determination inside the corridor (thin lines) obtained by A* path finding algorithm and quadtree space representation.

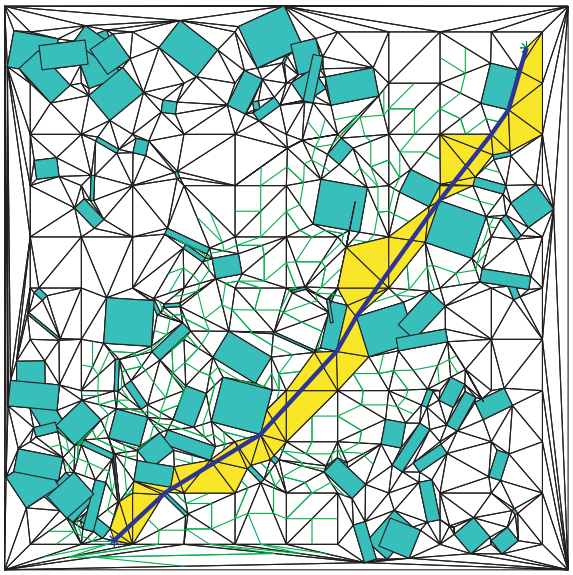

Fig. 4 Example of the optimal path (thick dark line) determination inside the corridor based on the triangular representation.

an object within the virtual environment would be unnatural in such cases.

This can be improved by considering the fact that the array of cells which are crossed by the calculated path actually defines a channel between the starting and the target point. This channel will be termed a path corridor and consists of a set of neigbouring square shaped cells when using quadtrees. When using CDT the corridor consists of a set of neighbouring triangular cells.

In both cases, the path within the corridor can be optimized by applying the funnel algorithm [5], which finds the shortest path within the corridor. The basic steps of the algorithm are briefly described as follows:

- The corridor is defined by two sets of points: a set of points defining the upper corridor border and a set of points defining the lower corridor border. In case of quadtree representation, the size of the neigbouring cells may differ, and in such a case only the overlapping part of the edges between the two cells is considered when defining lower and upper corridor border (see Figure 3). In case of triangulation the corridor border follows the edges of participating triangles.

- The start and the target point are linked to the upper and lower corridor border by additional edges.

- Let p be the starting point and let u and v be the points on the upper and lower border of the corridor, respectively. The shortest path from p to u and from p to v (not leaving the corridor) may overlap up to some point a. At a the paths diverge and are concave until they reach u and v. The a is called apex and the region delimited by path segments from a to u, a to v and uv segment is the funnel.

- The algorithm iteratively adds points on the corridor borders narrowing the funnel and discarding the points falling out of the narrowed funnel. When the top of the funnel shrinks down to a line, the shrunk part of the funnel defines a new segment of the shortest path and the new apex is set at the end of the new segment. The procedure stops when the target is reached. For more details, see [5].

For the example of Figure 1 the determined corridor and obtained shortest path within the corridor are given in Figure 3.

Similarly, the planned path from Figure 2 for the triangular representation is improved by funnel algorithm as shown in Figure 4.

2.4 Experimental results

The initial requirement of this investigation was to determine a path planning method which would enable fast, real-time path planning for moving objects within the virtual environments. The primary application area are various training simulators. Real-time in this case means a time, which is short enough not to be perceived by a human as an unrealistic behaviour of the moving object. Furthermore, we limited our investigation to situation with static obstacles, e.g. buildings in the area which is crossed by autonomous vehicles.

All mentioned algorithms were implemented in Matlab m-code and both segmentation methods were tested in combination with environments of different complexities.

Tab. 1 Computation times - quadtrees

Environment size 10 x 10 m, maximal obstacle size 1 x 1 m									
N_{obst}	*MinDim* [m]	N_{cells}	t_{QT} [s]	t_{A^*} [ms]	σ_{A^*} [ms]	t_{CO} [ms]	σ_{CO} [ms]	t_{FU} [ms]	σ_{FU} [ms]
10	10/32	273	0.2	9	6	4	2	6	3
	10/64	577	0.5	11	6	2	1	4	2
	10/128	1301	1.0	12	11	2	1	4	2
20	10/32	405	0.4	12	7	4	2	6	3
	10/64	877	0.8	11	8	3	1	5	2
	10/128	2189	1.8	14	12	3	1	5	2
50	10/32	797	1.0	13	9	4	1	6	2
	10/64	1921	2.6	23	23	3	2	5	2
	10/128	4817	6.2	49	56	4	3	7	4
100	10/64	3345	6.0	47	37	5	3	9	5
	10/128	9577	16.7	83	114	4	3	7	4
Environment size 100 x 100 m, maximal obstacle size 5 x 5 m									
100	100/256	12265	25	196	192	5	2	8	4

Tab. 2 Computation times - triangulation

Environment size 10 x 10 m, maximal obstacle size 1 x 1 m									
N_{obst}	*MaxDim* [m]	N_{cells}	t_{QT} [s]	t_{A^*} [ms]	σ_{A^*} [ms]	t_{CO} [ms]	σ_{CO} [ms]	t_{FU} [ms]	σ_{FU} [ms]
10	/	82	0.2	35	22	/	/	4	7
	1.25	244	0.4	63	54	/	/	0	0
	0.67	594	1.3	147	169	/	/	0	0
	0.4	1434	6.6	369	350	/	/	14	6
20	/	168	0.3	53	36	/	/	6	8
	1.25	328	0.6	96	67	/	/	0	0
	0.67	678	1.7	160	145	/	/	6	8
	0.4	1518	6.0	465	447	/	/	10	8
50	/	432	0.9	107	36	/	/	6	8
	1.25	606	1.6	133	91	/	/	4	7
	0.67	956	3.3	348	280	/	/	8	8
	0.4	1796	10.1	797	539	/	/	12	7
100	/	958	3.5	199	88	/	/	6	8
	1.25	1076	4.0	217	136	/	/	8	8
	0.67	1426	6.2	348	188	/	/	6	8
	0.4	2266	13.8	687	513	/	/	10	8
Environment size 100 x 100 m, maximal obstacle size 5 x 5 m									
100	/	838	2.8	352	220	/	/	10	8

Tables 1 and 2 show results of a set of experiments with varying environment size and varying number of obstacles within the environment on a 2.4 GHz PC. Within the tables, the first column shows the number of obstacles, next is the min/max size of segmentation cells, N_{cells} is the number of cells after the segmentation, t_{QT} and t_{DT} are the segmentation computing times, and the remaining columns show computing times and corresponding standard deviation of path search algorithm, corridor boundaries calculation, and the funnel algorithm, respectively. The computing times were obtained by eight consequent runs of the path planning algorithm with different start and target points.

The obtained results show that both methods can be used for path planning in real-time for moderately complex environments provided the space segmentation is done in advance. So only the path search, corridor calculation, and funnel algorithm need to be computed in real-time.

The drawbacks of the two segmentation approaches are two-fold. On the one hand, the paths are not smooth but consist of straight line segments, and on the other hand, the resulting paths often follow the edges of the obstacles, which is not realistic. Therefore an alternative way of path planning was considered, which is described in the next section.

3 Fast marching method based path planning

The Fast marching method (FMM) is based on numerical analysis of viscous matter and is a method of numerical solving a particular form of nonlinear partial equation, i.e. an Eikonal equation.

Simplified, the method gives a description of wavefront propagation through nonhomogeneous medium, where the propagation is represented by a wavefront arrival time for every point in the space.

When the propagation velocity for a point in space is defined by F (which is always non-negative), the arrival time function T is determined by the solution of equation

$$|\nabla T|F = 1 \tag{1}$$

at given border conditions, i.e. at a condition of zero value of T in the starting point. If F depends only on space coordinates, the above equation is an Eikonal equation.

The numerical solution of the equation is based on space grid, approximation of the gradient by the values in the neighbouring points and an efficient strategy of point calculation order.

Figure 5 shows a point in a 2D grid and its neihbouring points. In 2 dimensions the gradient $|\nabla T(x,y)| = |\nabla T(i,j)|$, where $x = i\Delta x$ in $y = j\Delta y$ can be substituted by an approximation (2)

$$\begin{pmatrix} max(D_{ij}^{-x}T,0)^2 + min(D_{ij}^{+x}T,0)^2 + \\ max(D_{ij}^{-y}T,0)^2 + min(D_{ij}^{+y}T,0)^2 \end{pmatrix} = \frac{1}{F_{ij}^2} \tag{2}$$

where

$$D_{ij}^{-x}T = \frac{T_{i,j} - T_{i-1,j}}{\Delta x}$$
$$D_{ij}^{-y}T = \frac{T_{i,j} - T_{i,j-1}}{\Delta y} \tag{3}$$

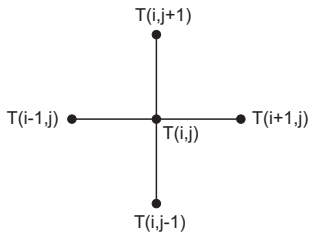

Fig. 5 A point in the grid and its neighbours

and

$$D_{ij}^{+x}T = \frac{T_{i+1,j} - T_{i,j}}{\Delta x}$$
$$D_{ij}^{+y}T = \frac{T_{i,j+1} - T_{i,j}}{\Delta y} \quad (4)$$

In [6] is proposed that the gradient is substituted by a simpler, less accurate approximation

$$max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2 = \frac{1}{F_{ij}^2} \quad (5)$$

Considering (3) and (4) the last equation can be modified to

$$max\left(\frac{T_{i,j} - min(T_{i-1,j}, T_{i+1,j})}{\Delta x}, 0\right)^2 + max\left(\frac{T_{i,j} - min(T_{i,j-1}, T_{i,j+1})}{\Delta y}, 0\right)^2 = \frac{1}{F_{ij}^2} \quad (6)$$

By setting new labels

$$\begin{aligned} T &= T_{i,j} \\ T_1 &= min(T_{i-1,j}, T_{i+1,j}) \\ T_2 &= min(T_{i,j-1}, T_{i,j+1}) \end{aligned} \quad (7)$$

the equation becomes

$$max\left(\frac{T - T_1}{\Delta x}, 0\right)^2 + max\left(\frac{T - T_2}{\Delta y}, 0\right)^2 = \frac{1}{F_{ij}^2} \quad (8)$$

Assuming F is always positive, T is monotonically increasing. The solution in a given point is only influenced by solution values in those points where the solution value is smaller. The fast marching method is based on the information propagation in one direction, from smaller values of T to larger values.

3.1 FMM method and path planning

The method can be easily applied to shortest path planning as the time of arrival into a point in the space is always the earliest possible time, and the known obstacles are simply taken into account during the calculation of the wavefront propagation. It suffices to set the propagation velocity to zero for any point inside the obstacle, which prevents wavefront from entering.

Once the arrival time function is calculated, the shortest path can be reconstructed by following the largest gradient. This can be done by simple Euler's method or by more precise Heun's method. The path can be calculated in both directions, i.e. from the wavefront starting point into any target point in the space or reversed, from a set of starting points to a fixed target point, which is the wavefront starting point. Such an example is shown in Figure 6.

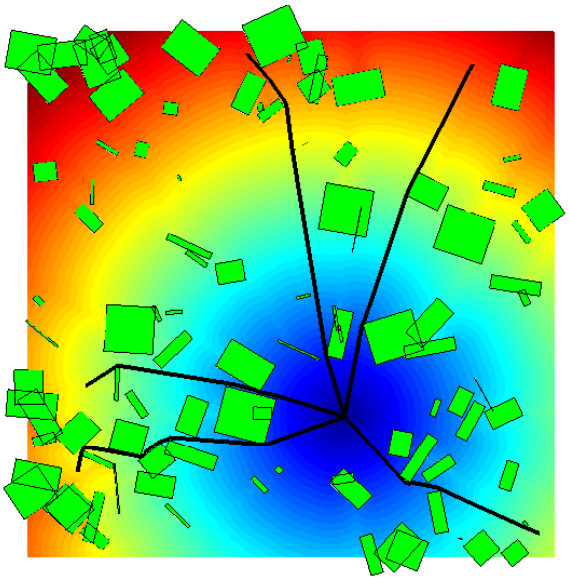

Fig. 6 Example of shortest path planning by FMM method and a fixed target point

3.2 Smooth path planning and FMM method

The arrival time function calculated by FMM method serves as a potential field which directs the moving object toward the target point. If additional information is included into this field, this can influence the calculated path. Such an information is the information about the distance to the obstacles. This way the path may be pushed away from the borders of the obstacles.

The distance to the obstacles can be included by the use of Extended Voronoi Transformation (EVT), which is used in digital image processing (and called Distance Transform therein). If EVT information is included into FMM based path planning, the obtained paths are driven away from the obstacles and smooth at the same time. An example is shown in Figure 7.

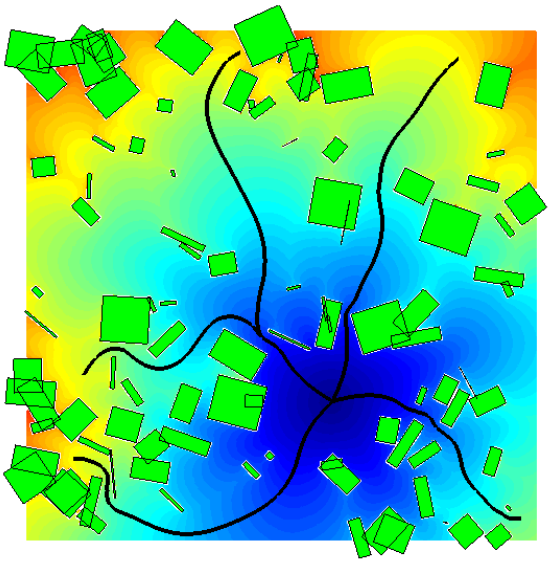

Fig. 7 Example of smooth path planning by FMM method and a fixed target point

4 The combined approach

While in general the FMM method gives better results in terms of optimality of the planned paths and can also be adapted to smooth down the paths as described above, its limitation in the substantial computational burden compared to segmentation based methods. E.g. in the example shown above the size of environment was 10 $\times$ 10 m, grid points were 0.02 m apart, the EVT calculation took approx. 600 s, but fortunately needs to be performed only once for a fixed configuration of obstacles. FMM algorithm takes around 30 s and then backtracking by Heun's method another 5 to 10 s, depending on the step size. All times were obtained on a 2.4 GHz PC by implementation of algorithms in Matlab m-code. The code is not optimal, nevertheless, the listed computation times indicate that the use of the method in real-time is not feasible for any complex environment.

The computational time for the calculation of the arrival time function strongly depends on the length of the propagating wavefront. It is relatively short when the wavefront can only advance in a narrow corridor. This implies the feasibility of a combined approach, where segmentation of the space is first applied to coarse initial path planning, then a suitable corridor is defined within which a finer path planning is performed by the FMM method.

The corridor used for FMM can be exactly the same that was used for the funnel algorithm. Given a set of obstacles, the EVT transformation is computed first. This could be done for points within the corridor only, but also the obstacles away from the corridor should be taken into account. Due to large computing time of the EVT it is more convenient to compute it for all the points in advance and then mask the points outside of the corridor by setting their EVT value to zero.

More precisely, the EVT transformation operates on a bitmap image and calculates the Euclidean distance from every pixel to the nearest pixel carrying a specific property, in our case a pixel that belongs to the obstacle. This is a concept related to Voronoi graph, with the difference that the information is attached to every picture element and not only the points on the edges of the Voronoi cells. A number of EVT algorithms calculate the Voronoi diagram as an intermediate step.

By using the EVT information during the FMM based path planning the planned path is automatically pushed away from obstacles and smoothed at the same time because the path does not follow the obstacle edges as it may happen with funnel algorithm.

In [3] the authors suggest to include the distance to the obstacles as a velocity parameter when calculating the wavefront propagation. This makes analogy to propagation of light ray through the medium with non-homogenous refractive index. The light in such a medium is not refracted but bent smoothly. For the path planning the distance is not used directly, but is transformed analogously to electric potential by a logarithmic function, e.g. [3]:

$$F_{i,j} = c_1 log(R_{i,j}) + c_2 \tag{9}$$

where $R_{i,j}$ stands for the distance of point (i, j) to the nearest obstacle. For the purpose of the presented study this function was further modified to

$$F_{i,j} = \begin{cases} log(\frac{R_{i,j}}{\Delta x}), & log(\frac{R_{i,j}}{\Delta x}) > 0 \\ 0, & log(\frac{R_{i,j}}{\Delta x}) <= 0 \end{cases} \tag{10}$$

Equality $\Delta x = \Delta y$ is assumed, and the proposed function enables balanced results with various distances of points in the grid. At the same time the function is consistent with the approach used by FMM, where the interior of obstacles is indicated by adjoining the corresponding grid elements by zero velocity of wavefront propagation.

With the described calculation of EVT the proposed path planning method can be summarized in the following steps:

- The QT or CDT segmentation of the environment with known obstacle positions is calculated.
- The environment is covered by a grid of points and the EVT transformation is calculated interpreting every point of the grid as a pixel of the image.
- A start and a target points are chosen and a corresponding path is planned by an A* type search algorithm. As a result, a path and a corresponding set of segmentation cells are obtained, the cells defining a corridor surrounding the calculated path.
- The information about the corridor boundaries is used to adjust EVT transformation by setting the calculated values of the EVT to zero for any point outside the corridor.
- The newly obtained EVT is used to parametrize the FMM based path search algorithm to obtain a smooth suboptimal path within the corridor.

If new path has to be calculated for another set of points, it suffices to repeat only the last three steps of the proposed procedure.

Some preliminary results regarding computational complexity of the proposed method are shown in Table 3 and 4. The first table shows the results obtained by the combination of quadtrees and FMM method while the second table shown the results obtained by the combination of constrained Delaunay triangulation and FMM method.

The number of obstacles N_{obst} is shown, the min/max size of segmentation cells, raster size used by FMM method, computing time of EVT transformation, and the remaining columns show the computation time needed for adjusting EVT to the given corridor (t_{EC}), for calculation of arrival time function within the corridor by FMM method (t_{FMM}), and for calculation of the path by following the largest gradient by Heun's method (t_H). By the last part, the integration step size was chosen according to the raster size: $step = 0.1/raster$

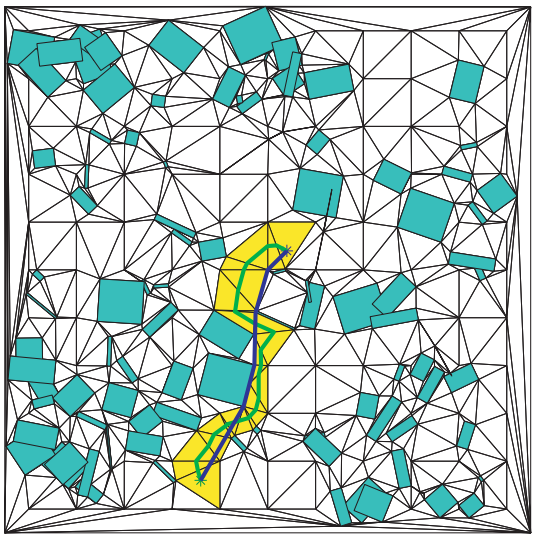

Fig. 8 Example of the path planning in triangular space segmentation – result of path search, the corridor and result of funnel algorithm.

Tab. 3 Combined method computation times - quadtrees

Environment size 10 x 10 m, maximal obstacle size 1 x 1 m						
N_{obst}	*MinDim* [m]	*raster* [m]	t_{EVT} [s]	t_{EC} [s]	t_{FMM} [s]	t_H [s]
100	10/128	0.02	612	9.4	1.0	21.5

Tab. 4 Combined method computation - triangulation

Environment size 10 x 10 m, maximal obstacle size 1 x 1 m						
N_{obst}	*MaxDim* [m]	*raster* [m]	t_{EVT} [s]	t_{EC} [s]	t_{FMM} [s]	t_H [s]
100	1	0.05	96	2.5	0.42	4.1
		0.02	612	12.1	2.6	22.5

5 Conclusions

The results show the feasibility of the proposed method and confirm the significant reduction in computation times of FMM in the corridor compared by the FMM for the whole search space. Due to rather small size of the minimal cells of the quadtree a small raster size must be chosen in this case to avoid numeric problems when following the gradient in the narrow parts of the corridor. This indicates that smaller granulation of quadtree is not necessary advantageous. Besides, the larger granulation yields more room in the corridor to FMM to smooth down the planned path. On the other hand, larger granulation prevents the path planning algorithm to draw the planned path over the narrow passages among obstacles. The resulting paths may therefore not be optimal in terms of their lengths.

Due to generally wider corridors obtained by triangulation compared to quadtree based path planning here also a larger raster size can be chosen, which significantly reduces the required computation times. The triangulation based path planning algorithm always tries to make the planned path shortest, even in cases where it has to pass narrow passages among obstacles. If the chosen raster is to large, however, the FMM algorithm may not succeed in calculating the time arrival function in such a passage.

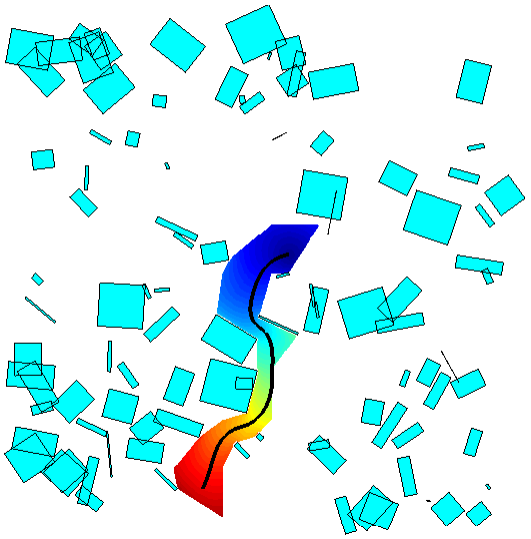

Fig. 9 Example of the path smoothing in triangular space segmentation – arrival time function within the corridor and resulting path obtained by the FMM algorithm).

6 References

[1] Baerentzen, J.A., On the Implementation of Fast marching Methods for 3D Lattices, Technical Report IMM-REP-2001-13, Technical University of Denmark, 2001.

[2] Farag, A.A., and M.S. Hassouna, Theoretical Foundations of Tracking Monotonically Advancing Fronts Using Fast Marching Level Set Method, Technical report, University of Louisville, 2005.

[3] Garrido, S., L. Moreno and D. Blanco, Exploration of 2D and 3D Environments using Voronoi Transform and Fast Marching Method, *Journal of Intelligent and Robotic Systems*, vol. 55, str. 55–80 (2009).

[4] Hongyang, Y, H. Wang, Y. Chen, D. Dai, Path Planning Based on Constrained Delaunay Triangulation, Proceedings of the 7th World Congress on Intelligent Control and Automation June 25 - 27,2008, Chongqing, China.

[5] Kallmann, M., Path Planning in Triangulations, Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland, July 31, 2005, 49-54.

[6] Sethian, J.A., Fast Marching Methods, *SIAM Review*, vol. 41, no. 2, str. 199–235 (1999).

[7] Botea, A., Mller, M., Schaeffer, J., Near optimal hierarchical path-finding, *Journal of Game Development*, vol. 1, pp. 7-28, (2004).

[8] Davis, I., Warp speed: Path planning for star trek: Armada, *AAAI Spring Symposium on AI and Interactive Entertainment*, AAAI Press, Menlo Park, CA., (2000).