

FAST ALGORITHM OF PLANNING A ROBOT'S PATH IN 2-DIMENSIONAL SPACE ON THE BASIS OF TRIANGULATION

PhD student Dmitry N. Aldoshkin, Assoc. Prof. Alexander N. Pupkov,

Assoc. Prof. Roman Yu. Tsarev

Siberian Federal University, **Russian Federation**

ABSTRACT

This article describes an approach to solution of a problem of planning a mobile robot's path in 2-dimentional space with obstacles. It gives the problem statement, which implies that there is no prior information about surrounding environment. It is supposed that the robot gathers real-time information via on-board sensors. The article also presents a theoretical analysis of such approach performance, along with comparison of the proposed approach to the existing ones, and demonstration of the suggested one's advantages. The simulation experiment results fully proving the theoretical thesis are also represented.

Keywords: mobile robot, path planning, path, pathfinding, triangulation

INTRODUCTION

In simultaneous localization and mapping of the environment, information about which is not available for the moment of the mobile robot's operation starts, the main tasks are mapping of the surrounding space, defining the robot's location in this space by the robot, and construction of the robot's path ensuring its free moving to the destination point. Solution of these problems presents a basis for solution of higher-level problems.

The task setting is the following: in the surrounding space $\Omega = E \bigcup K$ containing both obstacles $E = \bigcup_i E_i$ (i – number of obstacles), and free space $K = \bigcup_j K_j$ (j – number of mutually unreachable areas of free space), $E \bigcup K = \emptyset$, to find such path $M = \{M_i | M_i \in K, i = \overline{1, N}\}$ in the free space providing that the robot P moving from starting point M_1 to destination point M_N does not collide any obstacles at any point M_i of its path: $M \bigcap E = \emptyset$ (fig. 1).

At present there are many publications describing algorithms for the stated problem solution [1], [2], [3], [4], [5], [6], [7]. These algorithms can be divided into two groups:

- 1 algorithms not planning path – in this case, motion direction at each point of time (motion period) is calculated as some function from data, acquired by the surrounding space sensors [1], [2], [3];
- 2 path planning algorithms:
 - a) mapping data are abstracted by periodic structures (grids) [4], [5];
 - b) non-periodic graph formalizations of mapping data [6], [7].

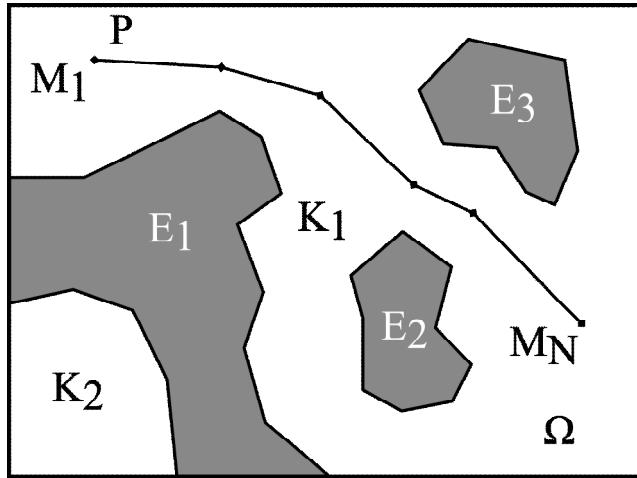


Fig. 1. Scheme of the space with obstacles and possible path of the robot

Algorithms of the first group can be applied for initial gathering of mapping information (in the beginning of work or navigation in yet unexplored area of the environment), and to avoid collisions with dynamic obstacles.

Algorithms of the second group use all available information about the surrounding space. Application of periodic grids demands making a compromise solution: high grid density allows better granulation/detail, but causes excessive cost for description of the environment, and vice versa— low grid detail result in decreasing detail of the space description. Algorithms using non-periodic graph formalizations lack such disadvantage – they allow minimizing the costs for the space description, while preserving control over detail.

The most general and widely used algorithm realizing the approach of non-periodic graph formalization is the algorithm, described by O'Rourke in [6]. This algorithm uses a graph of obstacle visibility as mapping data abstraction. For path finding problem solution on a graph, the generalized formulation of Dijkstra's algorithm is used, this enables to choose a suitable modification of algorithm to solve a certain problem with its limitations. Further in the article, the author will use O'Rourke's algorithm as a reference algorithm for both theoretical analysis and comparison of performance with the suggested algorithm.

FORMAL DESCRIPTION OF THE ALGORITHM

The purpose of the algorithm is to construct motion path M .

The source data is mapping data containing information about geometry of presently known surrounding space and information about geometry of the known obstacles.

The algorithm consists of the following steps:

- 1 Constructing a polygon of free space $P(K)$:
 - 1) Initializing an empty $P(K)$;

- 2) Constructing a contour of the surrounding space $C(\Omega)$ in the shape of a broken curve;
- 3) completing $P(K)$ with a broken $C(\Omega)$ as a curve limiting the polygon;
- 4) for each of the obstacles E_i in E :
 - a) constructing an obstacle's contour $C(E_i)$;
 - b) completing $P(K)$ with a broken $C(E_i)$ as a hole;
- 2 constructing a triangulation of the free space polygon $T = T(P(K))$;
- 3 constructing a dual graph of the triangulation $G(T)$;
- 4 defining the graph's nodes $G(T)$ corresponding to the current position and the target position;
- 5 defining a path W from the current position to the target one by pathfinding on the graph $W = \text{Dijkstra}(G(T))$;
- 6 altering the graph representation of path W into geometric representation M .

Further, separate steps of the algorithm are taken separately; the article describes limitations for source data, provides theoretical and experimental assessment of performance for their possible realizations.

SOURCE DATA

With no loss of generality, it is possible to suggest that ground mobile robots have got two degrees of freedom for movement, therefore the analysis of two-dimensional environment map is enough for the problem of path construction.

The source data for constructing the path of a mobile robot is the current data of the surrounding environment mapping, of a mobile robot's position in it (starting point), coordinates of an endpoint and geometrical parameters of the robot.

To be applicable for the suggested algorithm operation, the mapping data should be preliminarily processed and represented as a collection of the surrounding space objects' contours (taking into account the coordinates of its objects).

In the framework of the proposed algorithm, a representation form for these objects is limited by the following: any contour Π of an obstacle E shall be approximated by the broken curve, i. e. represents a collection of two sets: a set of vertices $\Sigma = \{\sigma_i, i = 1, V\}$ and a set of edges connecting the vertices $N = \{v_j, j = 1, R\}$: $\Pi = \{\Sigma, N\}$. The contours shall not have self-intersections: $v_i \bigcup v_j = \emptyset; i, j = 1, R; i \neq j$. Self-intersection of contours is not allowed for two reasons: a) the self-intersecting contours have not got a physical sense and indicate erroneous operation of the sensors data processing algorithms; b) the contours with self-intersections are not supported by a considerable number of triangulation algorithms due to ambiguousness of such contours interpretation.

The robot geometry can be represented in the form of:

- a point object;

- a limiting circle of non-zero radius;
- a limiting convex polyhedron;
- a polyhedron describing the robot's form in details.

The most often considered case is a limiting circle. This is due to the fact that it allows constructing a path which excludes collisions at relatively high performance level. Solution of the task with description of a robot as a point in its pure form is not applicable, however some simple transformations reduce the task with the description of a robot as a circle to this very task [6]. A complete solution of the task with application of representations in the form of a limiting convex polyhedron and a polyhedron precisely describing the robot's form is substantially more difficult task in its algorithmic sense, that is why it is not applied in practice due to limited computational performance of the mobile robot and excessive accuracy of the results.

Thus, O'Rourke's algorithm and the proposed algorithm both use point representation of a mobile robot geometry.

APPROACH TO GRAPH FORMALIZATION

The suggested algorithm for solution of a pathfinding task uses a graph abstraction of the space being explored. This approach to the environment description has been chosen because at present the instrument of graph analysis, including pathfinding, is increasingly developed and studied.

As it was stated in the above, both nodes of periodic grid and non-periodic elements characterizing the environment with the proper accuracy can serve as a graph's vertices.

Periodic grids have been excluded from consideration during earlier stages of the set problem analysis for the following reasons:

- the construction of a grid requires choosing a step, which is not a trivial task;
- periodic grids are crucially excessive: description of one physically continuous segment of space can require using several cells which, when transit to graph representation, expand the graph's size.

The reference algorithm suggests choosing vertices of the obstacles contours as nodes, and relations of visibility between the vertices (i. e. all sets of diagonals of the polygons in the free space) – as edges of the graph. Such formalization allows detecting a path from the starting point to the target one. Simultaneously, the given path shall be optimal in terms of geometrical length. The following can be referred as disadvantages of this algorithm:

- a big size of the graph: number of its vertices equals to a number N of all the vertices of all the obstacles' contours, while a number of edges of the graph reaches $\frac{N^2 - 3 \cdot N}{2}$ (the case of a convex polygon) at the worst, i.e. we can reckon that the number of the graph's edges (diagonals) can be esteemed as [6]:

$$E = O(N^2); \quad (1)$$

- a large number of cycles on the graph which imposes limitations to algorithms of graph traversal and path searching.

The consequence of large amount of information based on which we construct a graph is a greater algorithmic complexity of its construction. The best estimations are those of Welzl's algorithm ($O(N^2)$) [8] and algorithms of Ghosh and Mount ($O(N \log N + E)$) [6]. Complexity of the latter depends on a number of edges calculated as $E = O(N^2)$. Considering this, a complexity of the algorithm of visibility graph construction at the worst comes out at $O(N \log N + N^2)$. For this reason further we consider Welzl's algorithm.

As will be illustrated below, the stated task of pathfinding with no limitations to its optimality can be solved with drastically less computational costs. To achieve this, we need to choose such free space abstraction that would describe it comprehensively and would have a sufficiently smaller size than one on the reference algorithm. Let us select a triangulation of polygons of the free space as such abstraction. For each triangulation, there is a dual graph that further will be used for pathfinding. A dual graph is a graph, vertices of which correspond to the triangulation triangles, while its edges are relations of adjacency between the triangles (a shared edge).

A dual graph of triangulation, on the one hand, completely describes the free space structure and, on the other hand, it has a size less than the size of a graph derived from using the reference algorithm. A number of vertices of a dual graph is a value depending upon input data (a number of obstacles first of all), and amounts to $N + 2m - 2$, where m – number of obstacles. The number of vertices is $O(N)$, which corresponds to the one for the algorithm described above. A number of edges depends on input data as well and equals $N + 3m - 3$ (with evaluation of $O(N)$), which dramatically exceeds the evaluation (1).

Therefore, a graph's size in the framework of the suggested algorithm is sufficiently smaller in the sense of the worst esteems than in the reference algorithm. This fact is relevant in analysis of algorithm of pathfinding on graphs given hereinafter.

An equally important issue is a problem of algorithmic complexity of triangulation construction. There are many algorithms of triangulation construction with different complexity [6]. Evaluation of complexity for the best algorithm is $O(N)$, which substantially exceeds evaluation $O(N^2)$ in O'Rourke's algorithm for the pathfinding problem solution. However, a practical realization of such algorithm is increasingly difficult and slow, so algorithms with theoretically slightly lower performance are applied. In the framework of simulation experiment described below we use Siedel's algorithm with the complexity of $O(N \log^* N + k \log N)$ [9].

GRAPH SEARCH ALGORITHMS

At present the task of pathfinding on graphs studies outstandingly well. Dijkstra's algorithm for finding the shortest path on a graph should be noted in the first place among the proposed algorithms. Another widely known algorithm is A* search algorithm – a modification of Dijkstra's algorithm based on using heuristics [10]. A* generally outperforms Dijkstra's algorithm; however their worst-case performances correspond. A*'s tendency to exit to local minima with extensive stay there (which results in crucial decrease of performance in pathfinding tasks in the presence of a large

number of complex nonconvex obstacles) can be singled out as a substantial disadvantage of A*. By virtue of its heuristic nature, A* algorithm has a lot of modifications and variants like, for instance, WA*[11], HGA*[12], R*[13] and others.

In the framework of the suggested algorithm, we use Dijkstra's algorithm because it does not share this tendency to a prolonged stay in a local minimum. While the complexity evaluation for this algorithm performed with the use of Fibonacci heap amounts to [14]:

$$O(V \log V + E), \quad (2)$$

where V – number of the graph's vertices, E – number of edges.

Transition from Dijkstra's algorithm to A* (or its modifications) is not a problem, but it requires construction of heuristics specific for a concrete problem statement limitations. The issue of such heuristics construction is not a subject of this piece of work.

As it has been stated in the above, a number of vertices for visibility is $V = N$ (where N – a number of vertices of all the contours of the obstacles), and a number of visibility graph's edges is estimated as $E = O(N^2)$. In this case, the complexity (2) of Dijkstra's algorithm operation shall be $O(N \log N + N^2)$.

For a dual graph of triangulation, an estimated number of vertices is $V = O(N)$, and a number of edges equals $E = O(N)$, which gives the estimated performance $O(N \log N + N)$.

OVERALL COMPLEXITY

Pathfinding algorithm consists of two stages: construction of a free space graph and pathfinding on this graph. The complexity of this algorithm can be calculated as a sum of its stages complexity. Thus, it is obvious that the complexity of the algorithm described by O'Rourke is equal to $O(N \log N + N^2)$. The suggested algorithm's complexity is $O(N \log N + N)$. Experimental results given below confirm the assumption about the difference in the complexity of the algorithms.

SIMULATION EXPERIMENT

The aim of the simulation experiment is to validate the suggested algorithm and to demonstrate its high performance in comparison to the known ones.

As part of the experiment we compared operation of two algorithms: the one based on the visibility graph [6] (reference algorithm) and the proposed algorithm.

As it was mentioned earlier, Welzl's algorithm with the complexity of $O(N^2)$ was used as an algorithm of visibility graph construction. As there is no open realization of this algorithm, the author realized it using C++ programming language.

As realization of a triangulation algorithm we use Siedel's algorithm with the complexity $O(N \log^* N + k \log N)$, which feebly differs from a linear one with constant k due to drastically slow growth of the recursive logarithm. Realization of this algorithm is supported by *poly2tri*.

Dijkstra's algorithm has also been realized by the author because the existing open realizations are either increasingly generalized, or they docking with the above described algorithms realizations is complicated.

The simulation experiment was carried out on synthetic data, because it requires a great effort to prepare a set of real data with a definite step of the sets dimensions. While synthetic data are generated procedurally, and consequently control over dimension and other parameters of the sets is not a problem.

The experiment uses the following scene: a rectangular surrounding space Ω , where rows and columns enclose rectangular obstacles E_i (fig. 2). Minimal selection contains 2 columns and 1 row. A number of obstacles in each following set increases by 1 column and 1 row. Such approach, on the one hand, ensures a dimension step for the data sets which allows to demonstrate the pattern of operation time growth for one or another algorithm, but on the other hand, it allows not to spend much time for processing of the sets which are slightly different in size, and so quickly shift to analysis of larger sized sets.

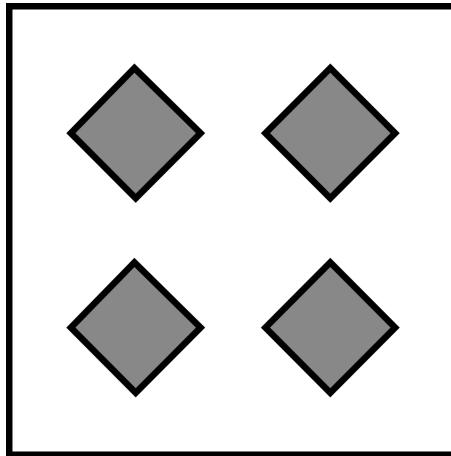


Fig. 2. A sample of the experimental scene. Grey blocks correspond to obstacles E_i ; white area is the surrounding space Ω

In the result of the experiment we got a row of patterns demonstrating substantial parameters of the algorithms operation, precisely:

- operation time of the first stage (construction of a graph) (fig. 3);
- size of a resultant graph in vertices and edges (fig. 4);
- operation time of the second stage (pathfinding on the graph) (fig. 5);
- total operation time (fig. 6).

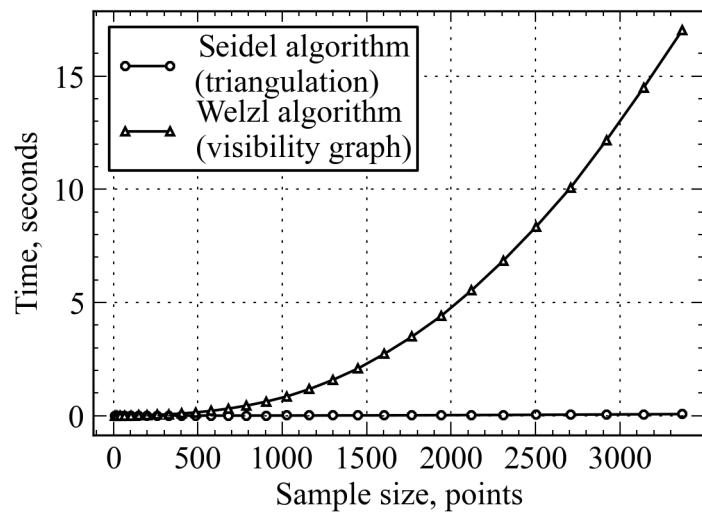


Fig. 3. Comparison of operation time of the first stage (construction of a graph)

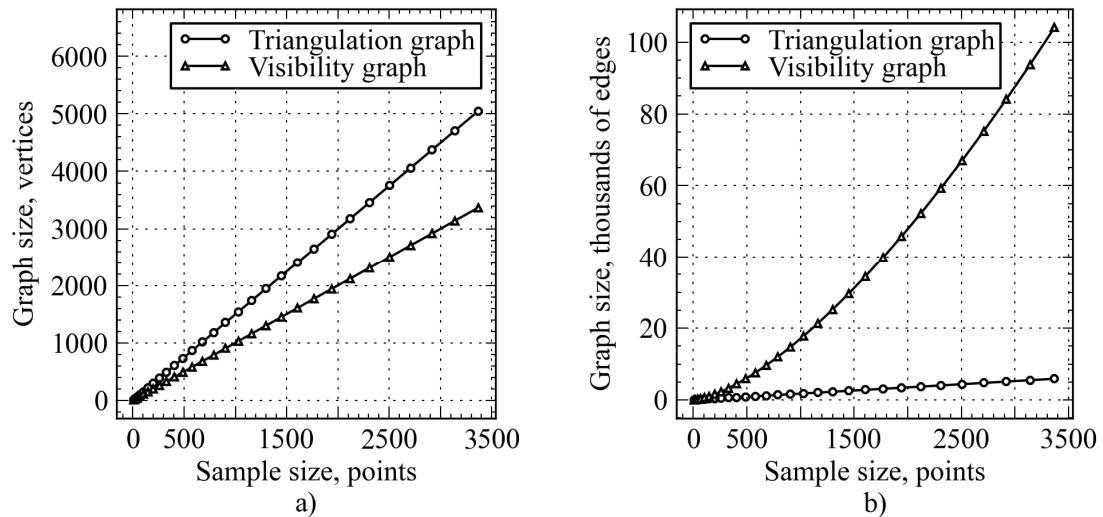


Fig. 4. Comparison of the graph size: a) vertices; b) edges

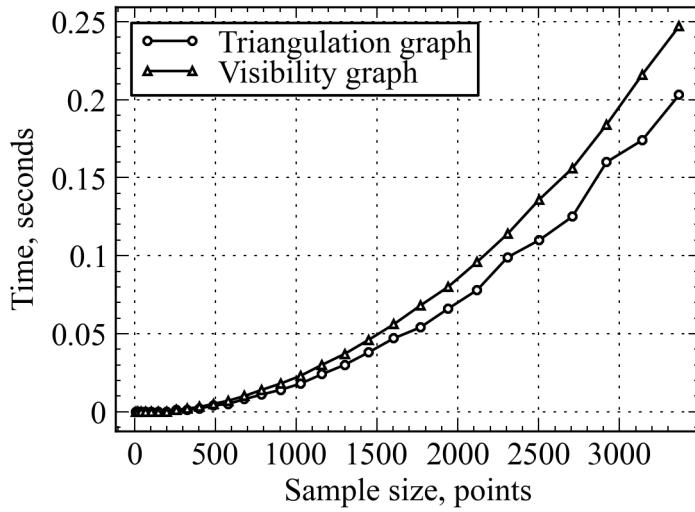


Fig. 5. Comparison of operation time of the second stage (pathfinding in the graph)

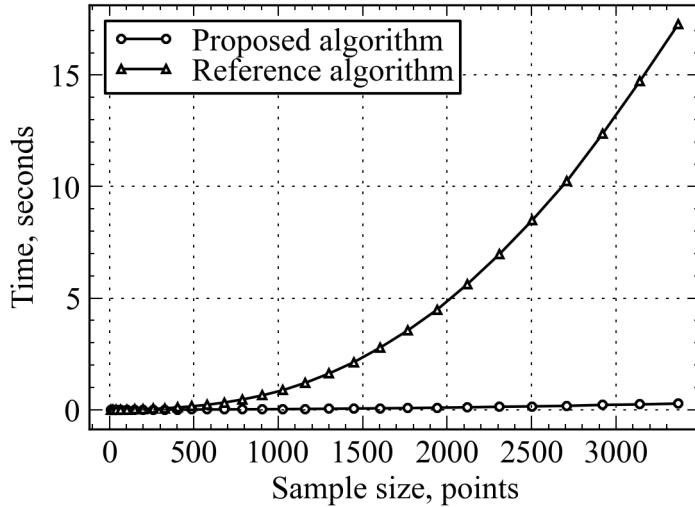


Fig. 6. Comparison of total operation time of the algorithms

On the basis of data provided in fig. 6, we can draw a conclusion about sufficient advantage in the suggested algorithm performance.

The suggested algorithm performance is conditioned by the fact that its first stage is executed faster than a corresponding stage in the reference algorithm and makes major contribution to a relative increase of its performance (fig. 3). Besides, the second stage is performed with less time consumption (fig. 5). Notwithstanding that the same algorithm (Dijkstra's algorithm) as in the reference algorithm is used at the second stage, still its execution requires less time. This results from the fact that the number of vertices of the graph influences the algorithm operation time more drastically than the number of edges (about which we can conclude from evaluation of complexity for

Dijkstra's algorithm): the number of vertices of the analyzed graph is substantially lower than one in the reference algorithm (fig. 4a), though the number of edges exceeds the number of edges in the reference algorithm (fig. 4b), which leads to less operation time (fig. 5).

Fig. 7 a, b allows to assess quality of the final solution. The suggested algorithm does not provide a solution optimal in terms of geometrical length of a path, but enables to construct a path, the length of which is optimal in terms of the graph abstraction's metrics.

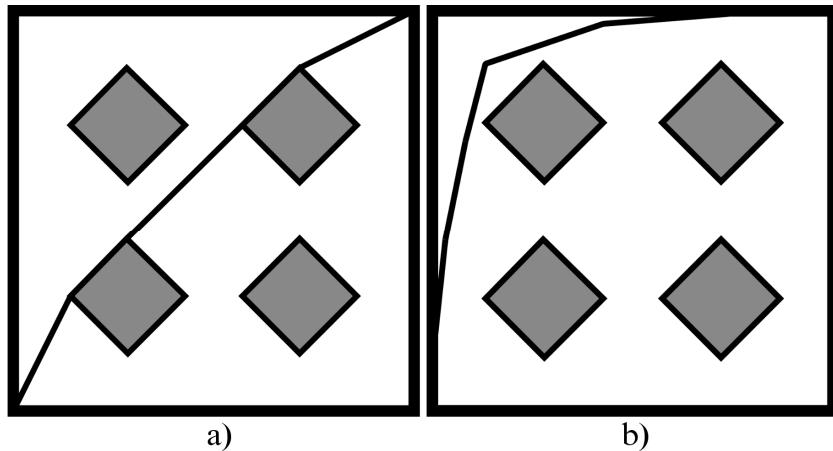


Fig. 7. Paths constructed according to: a) the reference algorithm; b) the proposed algorithm

In general case such path will be substantially different from an optimal one, however this difference can be minimized by a set of tools: at the expense of choosing some other geometrical elements of the triangulation grid as the graph's vertices; detection of convex polygons and consolidation of the correspondent triangles in a graph in a unified vertex; geometrical optimization of the resultant path.

CONCLUSION

This work suggests an algorithm for solution of a problem of pathfinding for a mobile robot in two-dimensional space with obstacles. It was shown that the suggested algorithm has got a substantial advantage in performance in comparison to the reference algorithm. This theoretical advantage was proved in practice by a simulation experiment, where separate stages of the suggested and the reference algorithms were compared.

However, the experiment showed that despite the advantage in performance, the suggested algorithm does not allow to achieve the solution quality as that of the reference algorithm.

Thus, the suggested algorithm is not capable to completely supersede the reference approach, however its application is reasonable in a number of cases: for definition the existence of a path between two points of space; for calculation of a motion path of a mobile robot, when time costs for solution construction are more crucial than the path optimality.

REFERENCES

- [1] Lu L., Gong D., Robot path planning in unknown environments using particle swarm optimization, 4th International Conference on Natural Computation, ICNC, 2008, China, vol. 4, pp. 422-426.
- [2] Rusdinar A., Kim J., Lee J., Kim S., Implementation of real-time positioning system using extended Kalman filter and artificial landmark on ceiling, Journal of Mechanical Science and Technology, vol. 26/issue 3, pp. 949-958, 2012.
- [3] Saranya C., Unnikrishnan M., Akbar Ali S., Sheela D.S., Lalithambika V.R., Occupancy grid based path planning and terrain mapping scheme for autonomous mobile robots, International Journal of Control Theory and Applications, vol. 8/issue 3, pp. 1053-1061, 2015.
- [4] Das P.K., Behera H.S., Pradhan S.K., Tripathy H.K., Jena P.K., A modified real time A* Algorithm and its performance analysis for improved path planning of mobile robot, Smart Innovation, Systems and Technologies, vol. 32, pp. 221-234, 2015.
- [5] Yakovlev K., Baskin E., Hramoin I., Grid-based angle-constrained path planning (2015) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9324, pp. 208-221, 2015.
- [6] O'Rourke J., Computational Geometry in C, 2nd ed., Cambridge University Press, England, 1998, 392 p.
- [7] Reuter M., Harneit S., Pathfinding with Kohonen-maps, 3rd International Industrial Simulation Conference, ISC, Germany, 2005, pp. 73-77.
- [8] O'Rourke J., Art gallery theorems and algorithms, Oxford University Press, New York, United States, 1987, 304 p.
- [9] Seidel R., A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, Computational Geometry: Theory and Applications, vol. 1/issue 1, pp. 51-64, 1991.
- [10] Hart P.E., Nilsson N.J., Raphael B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, vol. 4/issue 2, pp. 100-107, 1968.
- [11] Pohl I., First results on the effect of error in heuristic search, Annual Machine Intelligence Workshop, Edinburgh, Scotland, vol. 5, pp. 219-236, 1970.
- [12] Yakovlev K.S., HGA*, an efficient algorithm for path planning in a plane, Scientific and Technical Information Processing, vol. 37/issue 6, pp. 438-447, 2010.
- [13] Likhachev M., Stentz A., R* Search, National Conference on Artificial Intelligence, United States, vol. 1, 2008, pp. 344-350.
- [14] Levitin A., Introduction to the Design and Analysis of Algorithms, 3rd ed., Pearson, UK, 2012, 592 p.