UDC 519.8
MSC 90-08

Вестник СПбГУ. Прикладная математика. Информатика... 2023. Т. 19. Вып. 2

Dynamic path planning algorithm for autonomous mobile robot with a minimum number of turns in unknown environment

G. E. Rego, R. V. Voronov

Petrozavodsk State University, 33, Lenina pr., Petrozavodsk,
185910, Russian Federation

For citation: Rego G. E., Voronov R. V. Dynamic path planning algorithm for autonomous mobile robot with a minimum number of turns in unknown environment. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2023, vol. 19, iss. 2, pp. 264–274. https://doi.org/10.21638/11701/spbu10.2023.211

The article is devoted to the problem of reactive navigation of a mobile robot with limited information about the environment. An algorithm for finding a path from source to the target with a minimum number of turns is described. The idea of the algorithm is based on the bug family of algorithms for reactive navigation. The mobile robot remembers the boundaries of obstacles and calculates the angle of rotation depending on the surrounding situation. The difference from bug algorithms is that the robot does not move "along the obstacle", but turns only in a limited number of cases. The results of testing the algorithm on simulated polygons are presented. Models of fallen trees, stumps and swamps were considered as obstacles. The performance of the algorithm is evaluated by comparing the minimum possible number of turns with the number of turns in the path obtained using the algorithm.

Keywords: path planning, mobile robot, reactive navigation.

1. Introduction. The development of navigation algorithms is one of the key areas in modern scientific research, which is motivated by the active development of unmanned vehicles. In order for autonomous systems, such as unmanned aerial vehicles, cars or robots, to move independently, it is necessary to provide them with the ability to navigate in space [1]. This means that they must be able to determine their location, the location of the target and obtain enough information about the environment for route planning [2].

There are several approaches to the navigation of mobile robots. Usually there are 2 types of navigation: global planning and local planning. Sometimes reactive navigation is defined separately. Global navigation is characterized by the presence of complete information from the robot about the map of the area, traffic rules, as well as possible routes and communication with orbiting satellites. This type of navigation usually has a high computational complexity and gives predictable results. Global planning is used in unmanned vehicles. The map of the area is preloaded into the vehicle's memory, and the sensors detect the current situation [3–5].

Local planning differs from global planning in that not all information about the environment is available. In such cases, the robot only knows what it was able to detect using the sensors. Simultaneous Localization and Mapping (SLAM) [6–8] algorithms are based on local planning. The robot moves through an initially unknown area, and maps the objects encountered. There are many varieties of SLAM depending on sensors (visual [9], lidar [10]) or on the subject area (underwater [11]). Reactive navigation differs from other types in the small amount of memory available to the robot and the weak capabilities of its sensors. The robot at the initial moment of time knows only its source and target.

One of the most popular reactive navigation algorithms with proven convergence are bug algorithms [12, 13]. These algorithms use various metrics to evaluate their performance. However, none of them use the number of turns (or links of the path) taken by the robot as a performance metric. This article presents algorithms that are designed to minimize the number of links into which the robot's path from the source to the target is divided.

Such a metric will be useful when building routes in a forest area. For example, there is a forest robot whose purpose is to plant trees or carry out forestry work [14]. When carrying out work on felling forests, portages are laid. However, since felling is carried out by heavy equipment, and some time, up to several years, may elapse between tree felling and planting, we assume that the forest robot will move in the absence of roads. Usually, the conditions for performing work in a cutting area are determined by the type of soil in the area. As a rule, soil types are classified into four categories [15, 16]. The permissible duration of forestry work depends on the category of soil. The general principle is simple: the harder the soil, the longer the work can be carried out.

During the passage of machinery, the top fertile soil layer is broken. Especially during periods of high humidity and soil erosion (for soils of the second and third categories). Severe damage occurs in places where forestry machines turn. It is necessary to minimize the number of turns during the movement of forestry machines and a forest robot in order to minimize the detrimental effect on the fertile soil layer. At the same time, a typical cutting area contains many obstacles that need to be avoided. The most common of these are fallen trees, stumps and impassable places, such as swamps.

These factors actualize the task of developing a reactive navigation algorithm with minimization of the number of turns. The article has the following structure. Further the mathematical model of the problem being solved and formalizes the concepts used in the subject area is described. Then an algorithm for searching for a target point in an unknown terrain is presented, as well as an auxiliary algorithm for determining motion vectors in a collision with obstacles. After that the results of the algorithms is presented.

2. Mathematical model. Let's set the scene along which the robot model moves (hereinafter referred to as the robot) as a polygon P. A polygon is a part of the plane bounded by a closed polyline without self-intersections. The polygon may contain some restricted areas (obstacles). The set of obstacles will be denoted as $O = \{o_1, o_2, \ldots, o_P\}$. Each obstacle is a polygon. A link is a part of a straight line bounded by two points. The obstacles do not touch each other. This means that for any $o_a, o_b \in O, a \neq b, o_a \cap o_b = \emptyset$. The set of all points that belong to obstacles will be denoted as $\overline{O} = \cup_P O_p$.

The position of the robot R will be set as $p_r = (x_r, y_r, \theta_r)$, where x_r, y_r are the coordinates of the point where the robot is located, θ_r is the orientation of the robot (in degrees) relative to the target. At the moment of initialization of the solution to the problem, R is at the point $p_s = (x_s, y_s, 0) \in P$, where 0 means that the robot is directed to the target. The target $p_t = (x_t, y_t, 0) \in P$ is the point where the robot needs to get to. At each moment of time, a segment (p_r, p_t) is built between the current location of the robot and the target. θ_r is defined as the angle the robot must turn to be directed at the target.

Initially, the robot has no information about the map P. The robot has some visibility area V in the form of a circle of radius v. We define the current visibility area $V(p_r)$ as a set of points of the polygon P such that the distance between them and the robot is less than or equal to v and the segment between this point and the point where the robot is located does not intersect with any of the obstacles. Formally, this definition looks like this: $V(p_r) = \{p \in P : d(p, p_r) \leqslant v \ \& \ (p, p_r) \cap \overline{O} = \emptyset\}$. Further, we will write about this

area that the robot "sees" it. By free space in a given direction we mean the absence of obstacles on the trajectory of movement in this direction. If free space in the direction of movement is not less than v, then the robot tries to move the distance $v + \delta$, where δ is some very small number.

The total path traveled by the robot can be represented as a set of links $L = \{l_1, l_2, \dots, l_n\}$, where the index is the number of the link in the order of priority. Each link l_i can be represented as a part of a straight line bounded by a pair of points (p_{i1}, p_{i2}), where p_{i1} are the coordinates of the beginning of the link, p_{i2} are the coordinates of the end of the link. Neighboring links do not lie on the same straight line. At points that belong simultaneously to two adjacent links, the robot turns.

Let W be the set of all paths L from the source p_s to the target p_t, all links of which lie inside the polygon P and have no intersections with obstacles from the set O. We consider the path L_i better than the path L_j if $|L_i| < |L_j|$, where $|L_i|$ denotes the number of links in the path L_i. The path containing the minimum number of links will be denoted as L^*, that is, for any L_i from W such that $|L^*| \leqslant |L_i|$. Thus, the formal statement of the problem is reduced to solving the optimization problem:

$$\text{find a path } L \text{ from } W \text{ such that } |L| \to \min.$$

3. Heuristic algorithm for solving the problem of finding a path with a minimum number of turns. To solve the problem, we will put forward the assumption that the robot is able not only to see some space within a radius v from itself, but also to map the terrain (obstacle boundaries), as well as to estimate the distance to the visible object. By mapping we will understand the ability of the robot to map the area in which it has already been. However, unlike SLAM algorithms, the robot does not map everything it sees, but only the boundaries of the obstacles that the current direction rests against.

An arc between two points on a circle of radius v will be denoted as $\overset{\frown}{p_1 p_2}$. The set of collision points with an obstacle will be denoted as $H = \{h_1, h_2, \dots\}$, where $h_i = (x_i, y_i, O_i)$. From each point h_i, the robot can move in two directions along the obstacles α_{i1} and α_{i2}. The set of directions will be denoted as the set of vectors $\alpha = \{\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}, \dots\}$, where the first index is the index of the corresponding collision point from H, the second index is the numbering of the direction relative to the corresponding collision point, and the value itself $\alpha = \vec{p_1 p_2}$, where p_1 is the point at which the direction of movement is determined, p_2 is the point to which this direction leads. In the future, under α we will mean a vector, in the case, when the direction is determined, and a segment, in the case, when we are talking about a set of points and its intersection with other objects. Let us describe the formation of vectors in more detail.

When an intersection occurs between the current trajectory and some obstacle $O_{\text{cur}}, h_{\text{new}}$ is added to H. The robot stops at point p_r such that: $d(p_r, h_{\text{new}}) = v/2$. Directions of motion from the collision point are found using the helper algorithm described below. For definiteness, we will assume that we determine the direction α_{i1}. To determine α_{i2}, similar actions are performed with a deviation to the right from the collision point.

4. Algorithm for determining the direction of movement from the collision point.

Step 0. $p_1 = p_r$. It is necessary to determine p_2. We set the point p_0 on the circle of radius v as a point lying on the current trajectory of the robot. Set the deviation value ε (in degrees).

Step 1. Find the point p_{f_1} nearest to the left (by the angle of rotation) to p_0 such that $(p_r, p_{f_1}) \cup \overline{O} = \varnothing \ \& \ d(p_r, p_{f_1}) = v$.

Step 2. Deviate from p_{f_1} by ε to the left along a circle with center p_r of radius v. Let us denote the obtained point by p_{f_2}. If $(p_r, p_{f_2}) \cap \overline{O} = \emptyset$ then $p_2 = p_{f_2}$, the end. Else go to Step 3.

Step 3. Find p_{f_3} from $p_{f_1}\widehat{p}_{f_2}$ such that $(p_r, p_{f_3}) \cap \overline{O} = \emptyset$ & $d(p_{f_3}, p_{f_1}) \to \max$, $p_2 = p_{f_3}$, the end. The set of possible directions is formed based on the presence of obstacles around the robot. If there is free space around the robot for 360° and the current direction is not set, then there is only one direction: to the target. New directions appear at the moment when the current trajectory of movement intersects with some obstacle at a distance of no more than v from the robot. Other obstacles do not generate collision points if the current trajectory does not intersect with them. This situation is illustrated in Figure 1.

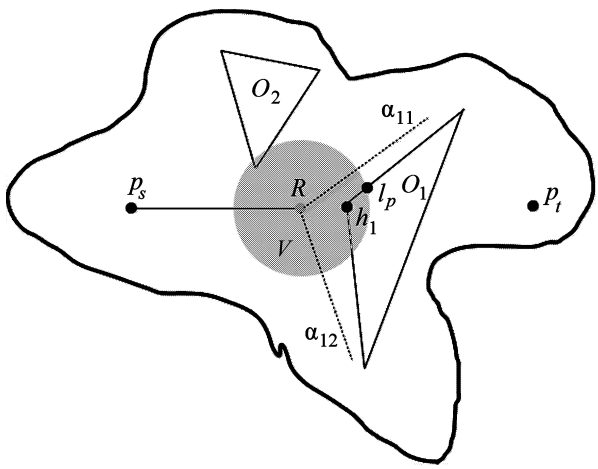

Figure 1. Determination of directions of movement when a collision point appears

A situation is possible when the robot, moving along one direction, also "sees" another (this is possible when the directions of two collision points of different obstacles are in the visibility zone of radius v). Formally $V(p_r) \cap \alpha_{ij} \neq \emptyset$ & $V(p_r) \cap \alpha_{mk} \neq \emptyset, \alpha_{ij} \neq \alpha_{mk}$, the direction α_{ij} corresponds to the collision point h_i and the direction α_{mk} corresponds to the collision point h_m. In this case, it passes simultaneously in two directions if the angle between α_{ij} and α_{mk} is acute and there exists $V(p_{r1})$ such that $h_m \in V(p_{r1})$ and $V(p_{r2})$ such that $h_i \in V(p_{r2})$ on the current trajectory. This situation is shown in Figure 2.

In the case when one of the obstacles O_{cur} is out of visibility area $V(p_r) \cap O_{\text{cur}} = \emptyset$, the last intersection point H_{new} is added to H. Formally $h_{\text{new}} = V(p_r) \cap O_{\text{cur}}$. The current direction corresponding to h_{cur} is marked explored.

We will call a dead end a situation when the robot ran into an obstacle and the only available direction of movement is where the robot came from to the current point. If the robot moved along two directions simultaneously and reached a dead end, then both directions are removed from the set A. When the robot has chosen some direction of movement α_{cur}, based on the collision point h_{cur}, this direction is removed from the set A.

5. Heuristic algorithm for finding a target in a maze with minimizing the number of robot turns.

Step 0. The parameters of the algorithm are determined, such as the source p_s, the target p_t, the radius of the scope v, and the polygon map P. The set of collision points $H = \emptyset$ and the set of directions $A = \emptyset$ are initialized.

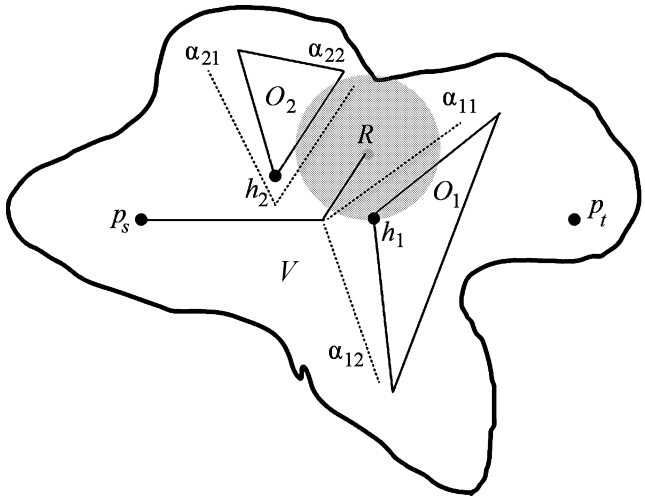

Figure 2. Movement along two directions (from different collision points) simultaneously

Step 1. The current segment (p_r, p_t) and the current scope $V(p_r)$ are initialized. If $(p_r, p_t) \cap V(p_r) \cap \overline{O} = \emptyset$, the robot moves along (p_r, p_t) until it reaches the target or an obstacle is encountered in the robot's path. If $p_r = p_t$ then end. If $(p_r, p_t) \cap V(p_r) \cap \overline{O} \neq \emptyset$ then go to Step 2.

Step 2. If $h_{\text{cur}} \in H$, then α_{cur}, which leads to h_{cur} and corresponds to h_{cur}, is removed from A, else h_{cur} is added to H. The set of directions of A is also completed. If α_{cur} corresponding to $h_{\text{cur}} = \emptyset$ then go to Step 5. Else $\alpha_{\text{cur}} = \alpha_{\text{cur1}}$ OR $\alpha_{\text{cur}} = \alpha_{\text{cur2}}$.

Step 3. The robot moves in the chosen direction along the obstacle until $(p_r, p_t) \cap V(p_r) \cap O_{\text{cur}} \neq \emptyset$ OR h_{new} occurs in the direction of movement. If $p_t \in V(p_r)$, then turn to the target and reach it. The end. If $(p_r, p_t) \cap V(p_r) \cap O_{\text{cur}} = \emptyset$ then go to Step 4. If $h_{\text{new}} \neq \emptyset$ then go to Step 2.

Step 4. The last intersection point (p_r, p_t) and O_{cur} that the robot has seen is fixed and h_{new} is stored: $(p_r, p_t) \cap V(p_r) \cap O_{\text{cur}} = h_{\text{new}}$. It checks if the target is inside O_{cur}. Next, the robot travels some short distance (for example, $v/10$) along the same trajectory. If $(p_r, p_t) \cap V(p_r) \cap \overline{O} = \emptyset$, then go to Step 1, else the robot fixes the collision point with the obstacle on the way to the target as h_{new}, go to Step 2.

Step 5. If $A = \emptyset$ then the end (the problem has no solution), else the robot chooses the nearest (by the number of turns) collision point h_i that has at least one unexplored direction $(d(p_r, h_i)) \to \min$ & $\alpha_{i1}, \alpha_{i2} \cap A \neq \emptyset$ and moves towards it. The movement is carried out along the already explored area, so for it you can use the algorithm for finding the minimum path by the number of links on the polygon [17]. Go to Step 3. The end.

6. Convergence of the algorithm.

Statement. *The problem of finding a path with a minimum number of turns has no solution iff, in the algorithm for finding a goal in a maze with minimizing the number of robot turns, there comes a moment when at Step 5 the set of directions of movement* $A = \emptyset$ *and the set of collision points* $H \neq \emptyset$.

P r o o f. **Necessity.** Let the problem of finding a path with a minimum number of turns have no solution. Let us show that the presented goal search algorithm will lead to the state when $A = \emptyset$ and $H \neq \emptyset$.

Let us assume that there is one obstacle O_1 between the source and the obstacle.

Then the robot, having reached the collision point h_1, $H = \{h_1\}, A = \{\alpha_{11}, \alpha_{12}\}$, chooses one of the directions. Moving along the chosen direction, the robot maps the area. Thus, when $(p_r, p_t) \cap V(p_r) \cap O_1 \neq \emptyset$, an obstacle map will appear on some segment. Since $h_{\text{new}} = V(p_r) \cap O_1$ is fixed as a collision point, there will be collision points at the ends of the studied segment O_1.

We assume that for each iteration of the algorithm, the robot maps a part of the obstacle no less than ε, where ε is some small non-zero positive number. Therefore, each time moving along the selected direction from the collision point, the robot learns a new area of the obstacle and fixes the collision points at the ends of this area. This implies the finiteness of the algorithm. Since the algorithm searches for a target as long as there is at least one direction, and knowing that the perimeter of the obstacles is finite, the boundaries of a given obstacle will be completely explored in a finite number of iterations P_{O_1}/ε, where P_{O_1} is the perimeter of the obstacle.

There are two possible cases. In the first case, there is one (p_r, p_t) that does not intersect with an obstacle, then the goal will be achieved. In this case, at the time of setting the last direction, $A \neq \emptyset$. In the second case, the target is inside the obstacle, therefore, it is unreachable. The same reasoning can be extended to any finite number of obstacles. The total number of iterations for which a complete obstacle map will be obtained: $(\sum_o P_o)/\varepsilon$ (the sum of the perimeters of all obstacles divided by ε). As a result, either the target is inside some of the obstacles (thus $A = \emptyset$ and $H \neq \emptyset$), or there is one (p_r, p_t) that does not intersect with any of the obstacles.

Sufficiency. Let us prove the sufficiency of the conditions by contradiction. Let $A = \emptyset, H \neq \emptyset$ and the solution exists. This means that there is some kind of obstacle O' such that the robot, acting according to the algorithm, built its complete map. By condition, the target point does not lie inside this obstacle. Therefore, there is some point p lying on the boundary of O' such that the segment $(p, p_t) \cap O' = p$. Since when moving, the direction of movement of the intersection of the vector (p_r, p_t) with obstacles is checked, when mapping this point, the robot will change direction and begin to move towards the target. Further, either the goal will be reached, or a collision will occur with another obstacle O_i, at the boundary of which there is also p such that $(p, p_t) \cap O_i = p$. In view of the finiteness of the number of obstacles, sooner or later there will be a direction that will lead to the goal. We arrive at a contradiction with the fact that $A = \emptyset$.

7. Checking if the target is inside an obstacle. In order to check if the target is inside the current obstacle, it is necessary to classify the collision points according to their correspondence to a particular obstacle. To do this, we introduce the set of obstacles detected by the robot $O_f = \{o_{f_1}, o_{f_2}, \dots\}$. For each detected obstacle, a set of collision points $H_{f_{\text{cur}}}$ is formed from the set H. Each collision point h corresponds to some obstacle $O_{f_{\text{cur}}}$ from O_f, and the only one. Denote the first point of collision with $O_{f_{\text{cur}}}$ as $h_{0\text{cur}}$. The remaining collision points $\{h\}$ belonging to $O_{f\text{cur}}$ are numbered according to the distance (in links) from h_0. After h_{cur} is added to $H_{f_{\text{cur}}}$, a check is made to see if the target is inside the current obstacle. The set of points belonging to the current obstacle form a closed polyline $(P_o : \{h_{li}, h_{li+1}\} \in O_{f_{\text{cur}}}$ & $\{h_{rj}, h_{rj+1}\} \in O_{f_{\text{cur}}}$, where l and r are the left and right directions from h_0). The last collision points on the left and on the right are interconnected $(h_{li}, h_{rj}) : h_{li+1}, h_{rj+1}$ does not exist. If one of the branches is missing, then the connection is made with h_0. The target is unattainable under two conditions. First, the target must belong to the resulting polygon (p_t of P_o). Secondly, all points of P_o faces must be explored (mapped) earlier. If the conditions are met, the algorithm terminates due to the impossibility of achieving the target. If p_t is not from P_o,

then it cannot be said that the target is unattainable. If p_t is from P_o, but not all faces of P_o are mapped, then it is possible that p_t is not from O_{cur}. This situation is shown in Figure 3, *a*.

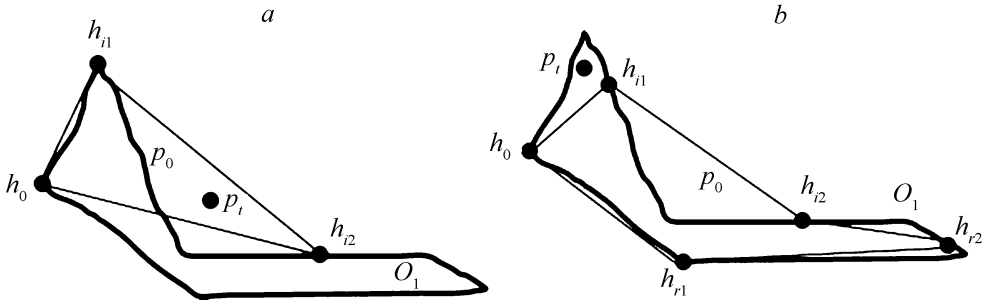

Figure 3. False positive (*a*) and negative (*b*) result of checking if the target point is inside the obstacle

Incompleteness. At the same time, a situation may arise when the target is inside an obstacle, but the procedure described above does not recognize it as such. An example is shown in Figure 3, *b*. Therefore, it is the absence of directions for research that is chosen as a reliable criterion for the absence of a solution to the problem.

Separately, it is worth considering the case when, from the point of view of geometry, one obstacle was classified as several. An example of such a situation is shown in Figure 4. In this case, the check is carried out separately for each obstacle from O_f. In a situation where obstacles are combined (the direction of one obstacle is opposite to the other), the construction of the combined polygon is performed as follows. Assume that the algorithm initially classified merged obstacles and collision point sets as O_{f_1} and the corresponding H_{f_1} : $\{h_{f_{10}}, h_{f_{1l_1}}, h_{f_{1l_2}}, \ldots, h_{f_{1r_1}}, h_{f_{1r_2}}, \ldots\}$, O_{f_2} and the corresponding H_{f_2} : $\{h_{f_{20}}, h_{f_{2l_1}}, h_{f_{2l_2}}, \ldots, h_{f_{2r_1}}, h_{f_{2r_2}}, \ldots\}$. For definiteness, let's assume that the obstacles merge on the segment $(h_{f_{1l_{\text{end}}}}, h_{f_{2r_{\text{end}}}})$. Then $O_{f_1+f_2}$: $\{h_{f_{1r_{\text{end}}}}, h_{f_{1r_{\text{end}-1}}}, \ldots, h_{f_{1r_1}}, h_{f_{10}}, h_{f_{1l_1}}, \ldots, h_{f_{1l_{\text{end}}}}, \ h_{f_{2r_{\text{end}}}}, h_{f_{2r_{\text{end}-1}}}, \ldots, \ h_{f_{2r_1}}, h_{f_{20}}, h_{f_{2l_1}}, \ldots, h_{f_{2l_{\text{end}}}}\}$.

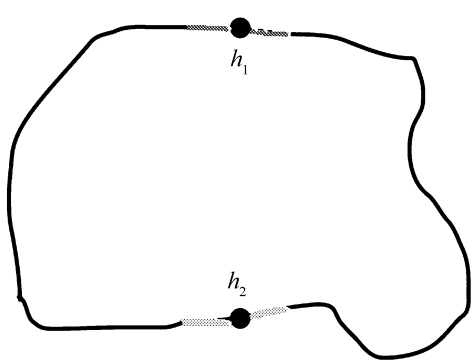

Figure 4. The mark h_1 — the first obstacle, the mark h_2 — the second (the robot identifies the obstacle as two different ones)

8. Experiments. The path L^* can be found by applying, for example, the algorithm for finding the minimum path in terms of the number of links [17]. However, the polygon map must be known for this. In practice, this is not always possible. The algorithm described above is applied precisely in such a situation when the map is not known. However, to measure the efficiency of the algorithm, when conducting experiments, we will compare the path L^* obtained using the algorithm [17] and the result $L^{\#}$ obtained using our algorithm.

For the experiments, objects were generated that have a similar shape to the real ones found in the forest area. Real objects that can be obstacles: a stump, a fallen tree, a swamp or any other impassable terrain. The model of a stump is a circle of small radius, the model of a swamp is a flat figure of arbitrary shape, the model of a fallen tree is a rectangle with a small length and a large width. Schematically, the obstacles are shown in Figure 5.

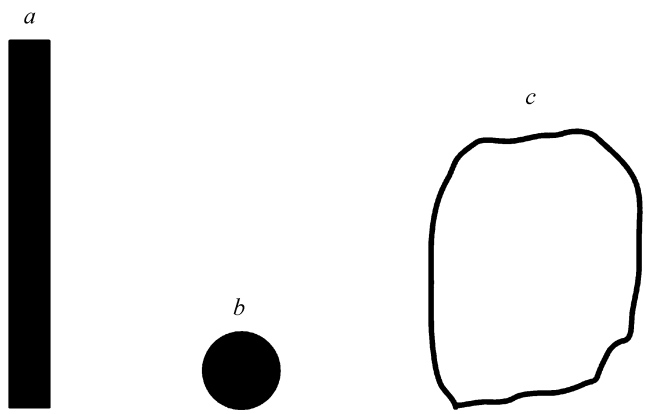

Figure 5. Typical obstacles in the cutting area
a — fallen tree; b — stump; c — swamp.

A performance metric was also calculated for each experiment. As a metric, we used the deviation of the obtained path length from the optimal one $e = (|L^{\#}| - |L^*|)/(|L^*|)$. The values of e lie between 0 and ∞. In addition, frequency histograms of the metric e were constructed. The histogram for experiment N 1 is shown in Figure 6.

The results of the experiments are presented in Table.

Table. **Results of experiments on finding a path on a polygon with a minimum number of links**

Number experiment	Number of links when applying the min-link-path algorithm	Number of obstacles	Obstacle type	Number of launches	Average number of legs	Median number of hops	Median e
1	2	1	Swamp	900	6.15	4	1.0
2	2	1	Stump	1000	4.38	4	1.0
3	2	1	Fallen tree	500	15.82	14	6.0
4	2	2	Swamps	900	6.39	5	1.5
5	2	5	Stumps	500	12.78	6	2.0

An example of how the minimum path in terms of the number of links and the path obtained using the presented algorithm look like is shown in Figure 7.

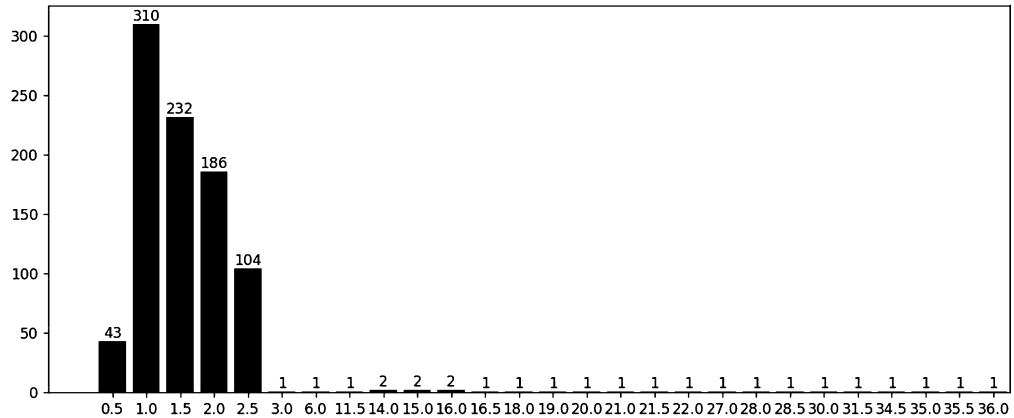

Figure 6. Histogram of the distribution of the results of experiment, on the x-axis, the metric e, on the y-axis, the number of measurements with this metric

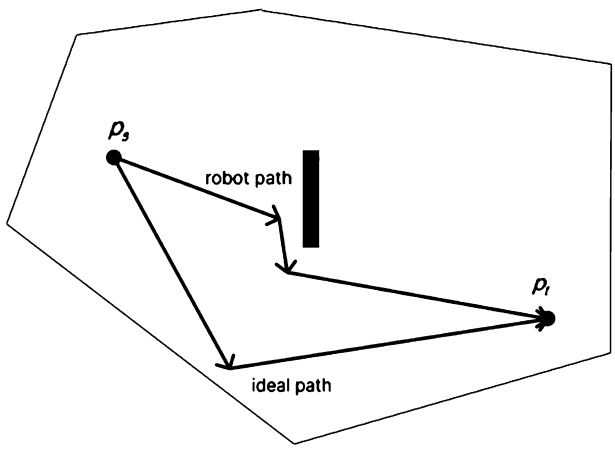

Figure 7. The difference between the minimum path and the path traveled by the robot

The experiments were carried out mainly on polygons with a small number of obstacles. In most experiments, the median e did not exceed two. This suggests that most often (with random generation of obstacles) it is possible to predict the length of the robot's route, approximately knowing the degree of terrain coverage by obstacles. In many ways, the efficiency of the algorithm depends on the selection of parameters. For example, if it is known that obstacles are mostly long (by long we mean obstacles whose length is several times greater than v), then it is necessary to increase the amount of movement along the current trajectory at Step 4 of the above algorithm.

9. Conclusion. This article discusses reactive navigation algorithms. The problems of this area are associated with the movement of a mobile robot in an unknown area in advance. This article considers a problem when the robot needs to get from the source point p_s to the target point p_t with a minimum number of turns. An algorithm for moving the robot around the polygon with minimization of the number of path segments has been developed.

A significant problem is the presence of obstacles. Existing algorithms assume that

the robot, after a collision with an obstacle, chooses the direction of movement, and then moves "along the obstacle". In this case, it is not clear how to count the number of path links.

The algorithm presented in this article solves this problem. The main idea of the algorithm is rectilinear movement after choosing a direction. This allows you to accurately count the number of links of the path and formalize the implementation of the turn. A separate algorithm is described for determining motion vectors. Its main idea is to move not directly to the last obstacle point that the robot sees at the current moment, but with some shift to the side. This is done in order to reduce the number of turns. It is assumed that the obstacle exceeds the visibility area and to bypass it, it is necessary to move to the side.

The presented algorithms were tested during experiments. Experiments have shown that the relationship between the radius of the field of view and the size of the obstacle is important. Determination of the optimal visibility area, in particular, for a forest robot requires a separate study.

References

1. Alatise M. B., Hancke G. P. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 2020, vol. 8, pp. 39830–39846.

2. Tang L., Yuta S. Indoor navigation for mobile robots using memorized omni-directional images and robot's motion. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, vol. 1, pp. 269–274.

3. Sukkarieh S., Nebot E. M., Durrant-Whyte H. F. A high integrity IMU/GPS navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, 1999, vol. 15, no. 3, pp. 572–578.

4. Pardhasaradhi B., Reddy Y. R., Cenkeramaddi L. R. Machine learning-based screening and measurement to measurement association for navigation in GNSS spoofing environment. *IEEE Sensors Journal*, 2022, vol. 22, no. 23, pp. 23423–23435.

5. Li X., Song B., Shen Z., Zhou Y., Lyu H., Qin Z. Consistent localization for autonomous robots with inter-vehicle GNSS information fusion. *IEEE Communications Letters*, 2022, pp. 120–124.

6. Smith R. C., Cheeseman P. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 1986, vol. 5, no. 4, pp. 56–68.

7. Smith R. C., Self M., Cheeseman P. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*. New York, Springer Publ., 1990, pp. 167–193.

8. Leonard J. J., Durrant-Whyte H. F. Simultaneous map building and localization for an autonomous mobile robot. *Proceedings of Intelligent Robots and Systems. Intelligence for Mechanical Systems*, 1991, vol. 3, pp. 1442–1447.

9. Chen Y., Zhou Y., Lv Q., Deveerasetty K. K. A review of V-SLAM. *IEEE International Conference on Information and Automation (ICIA)*, 2018, pp. 603–608.

10. Huang L. Review on LiDAR-based SLAM techniques. *International Conference on Signal Processing and Machine Learning (CONF-SPML)*, 2021, pp. 163–168.

11. Hidalgo F., Bräunl T. Review of underwater SLAM techniques. *6^{th} International Conference on Automation, Robotics and Applications (ICARA)*, 2015, pp. 306–311.

12. Lumelsky V. J., Stepanov A. A. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, 1986, vol. 31, pp. 1058–1063.

13. Ng J., Bräunl T. Performance comparison of bug navigation algorithms. *Journal of Intelligent and Robotic Systems*, 2007, vol. 50, no. 1, pp. 73–84.

14. Galaktionov O., Zavyalov S., Shchegoleva L., Korzun D. Features of building a forestry intelligent robotic system. *Proceedings of 29^{th} Conference of Open Innovations Association (FRUCT)*, 2021, pp. 433–436.

15. Schönauer M., Prinz R., Väätäinen K., Astrup R., Pszenny D., Lindeman H., Jaeger D. Spatio-temporal prediction of soil moisture using soil maps, topographic indices and SMAP retrievals. *International Journal of Applied Earth Observation and Geoinformation*, 2022, vol. 108, pp. 102730.

16. Galaktionov O. N., Kuznetsov A. V. Reduction of negative impact of skidders on the forest environment. *Astra Salvensis*, 2018, pp. 381–390.

17. Mitchell J. S. B., Rote G. Minimum-link paths among obstacles in the plane. *Proceedings of the Sixth annual Symposium on Computational Geometry*, 1990, pp. 63–72.

Received: February 14, 2023.
Accepted: April 25, 2023.

Authors' information:

Grigogij E. Rego — Postgraduate Student; regoGr@yandex.ru

Roman V. Voronov — PhD in Technical Sciences, Associate Professor; rvoronov@petrsu.ru

Алгоритм динамического планирования пути с минимальным числом поворотов мобильного робота при ограниченной информации об окружающей среде

Г. Э. Рего, Р. В. Воронов

Петрозаводский государственный университет, Российская Федерация,
185910, Петрозаводск, пр. Ленина, 33

Для цитирования: *Rego G. E., Voronov R. V.* Dynamic path planning algorithm for autonomous mobile robot with a minimum number of turns in unknown environment // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2023. Т. 19. Вып. 2. С. 264–274. https://doi.org/10.21638/11701/spbu10.2023.211

Статья посвящена проблеме реактивной навигации мобильного робота при ограниченной информации об окружающей среде. Описан алгоритм поиска пути из исходной точки в целевую с минимальным числом поворотов. Идея алгоритма основывается на семействе алгоритмов bug для реактивной навигации. Мобильный робот запоминает границы препятствий и подсчитывает угол поворота в зависимости от окружающей ситуации. Отличие от bug-алгоритмов заключается в том, что робот не двигается «вдоль препятствия», а поворачивает только в ограниченном числе случаев. Приводятся результаты апробации алгоритма на смоделированных полигонах. В качестве препятствий рассматривались модели поваленных деревьев, пней и болот. Работа алгоритма оценивается с помощью сравнения минимально возможного числа поворотов с числом поворотов пути, полученным с помощью алгоритма.

Ключевые слова: расчет пути, мобильный робот, реактивная навигация.

Контактная информация:

Рего Григорий Эйнович — аспирант; regoGr@yandex.ru

Воронов Роман Владимирович — д-р техн. наук, доц.; rvoronov@petrsu.com