Finding Time-Optimal Path Through a Forest of Circles by Graph Search

Vadim Belotelov[1] and Anna Daryina[1,2]

***Abstract*— Finding optimal path is required in many applications. However, there is no general method for doing it. There exist general numerical methods that optimize behavior of systems according to certain criteria, but these methods do not provide absolute optimum. In current article, a particular kinematic model of a wheeled mobile robot is concerned. For this model, it can be shown that the solution of the time-optimal problem of moving from one point to another through a forest of circular obstacles is one of the locally shortest paths between the two points, with addition of rotations around the starting and the finishing point. Therefore, an algorithm for automatic search of the optimal trajectory is proposed. It creates a graph of connections between the points on the circular obstacles and searches for the optimal trajectory on this graph using Dijkstra's approach.**

I. INTRODUCTION

The time-optimal problems with state constraints often appear in investigation of robotic motion. Although strict closed-form approaches are known for a long time, and the well-known Pontryagin's maximum principle [1] was first formulated in early 1960s, it is still the only approach that can provide closed-form solutions for optimal control problems. It has a number of limitations, one of them being inability to find optimal solution for multimodal problems (when there exist several locally optimal solutions). Therefore, for control problems numerical methods were developed, and a number of researchers contributed to application of numerical optimization methods to optimal control problems ([2], [3]). Methods that are applied to these problems include probabilistic roadmaps [4], rapidly-exploring random trees [5], nonlinear programming [6], metaheuristical optimization (genetic algorithms, particle swarm optimization, etc.) and many more, including even neural networks [7]. Metaheuristic optimization, while developed to find minimum (or maximum) of a function in parameter space of high dimension, can also be used to search for optimal control function by discretizing time and trying to find an optimal set of control parameters it the nodes [8]. Further, symbolic regression method has been developed to search for control function approximation in a closed-form [9].

Optimal control problem is not completely equivalent to multidimensional search of a minimum; not all numerical methods produce adequate search when it comes to optimal control; and evaluating right-hand side of a system model equation can be itself a time-consuming task, therefore there is still need of using other approaches, including further development of Pontryagin's maximum principle and extending its applicability to wider classes of systems [10], [11].

At the same time, many problems concerned with robotics use relatively simple models, for example, kinematic model of motion is used when we do not need to account for transients in the system and only investigate the motion as a sequence of steady-state motions. This article concerns time optimal problem for a wheeled mobile robot. In kinematic approach, when we assume that the wheel speeds can be controlled directly, it can be shown that the optimal trajectory is actually one of the the shortest paths between two points in (2-dimensional) space, and the control function is piecewise constant.

Besides, the results obtained in this article can be used for benchmarking numerical methods used to find optimal control functions.

II. MODEL USED AND PROBLEM STATEMENT

A. Model description

In the work, we use the kinematic model of a two-wheeled robot with independently driven wheels, known as differential wheeled robot (fig. 1). We assume that the body and the wheels of the robot are weightless, and we can independently control the speeds of each driving wheel.

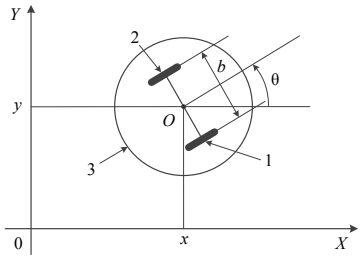

Fig. 1. Differential wheeled robot. 1, 2 – right and left wheels, 3 – body.

The mathematical model that describes the motion of such robot is as follows:

$$\begin{aligned} \dot{x} &= \frac{(u_1+u_2)}{2}\cos(\theta) \\ \dot{y} &= \frac{(u_1+u_2)}{2}\sin(\theta) \\ \dot{\theta} &= \frac{(u_1-u_2)}{b} \end{aligned} \tag{1}$$

where x and y are the Cartesian coordinates of the wheel pair center, and θ is the heading of the robot. We assume that

[1]Vadim Belotelov and Anna Daryina are with Federal Research Center "Computer Science and Control of the Russian Academy of Sciences, Russia, Moscow, Vavilova st. 44 b.2 vbelotelov@gmail.com

[2]Anna Daryina is also with the Department of Faculty of Computational Mathematics and Cybernetics of Moscow State University, Russia, Moscow, Leninskiye Gory, 2nd Study Block anna.daryina@mail.ru

the linear velocities of the right (1) and the left (2) driving wheels can be controlled directly, so the wheel radius is not considered. The single geometrical parameter b remains, denoting the wheel base of the robot.

The model has two independent control inputs, u_1 and u_2, denoting velocities of the right and left wheel, accordingly. We consider the following control limitation:

$$|u_i| \le u_{\max}, \quad i = 1, 2. \tag{2}$$

This model of a differentially driven robot is commonly used as a simple yet practical approximation in problems where there is no need to investigate precise transient dynamics of the robot, for example, when modeling a robot with high-torque motors, in SLAM (simultaneous navigation and mapping) tasks, for automatic delivery robots, etc. The geometrical dimensions of the robot can be neglected, and the robot is then considered a point. The distance b between the wheels is still required as a parameter, because it indicates the relation between linear and rotational movement of the robot when the control inputs are applied.

B. Time-optimal problem statement

Assume we have model (1) with control limitations (2). Let at the initial moment $t^0 = 0$ the state values are set as

$$x(0) = x_0; \quad y(0) = y_0; \quad \theta(0) = \theta_0, \tag{3}$$

and the task is to move to the terminal state defined as

$$x(t^f) = x^f; \quad y(t^f) = y^f; \quad \theta(t^f) = \theta^f, \tag{4}$$

in the least possible time t^f.

The robot moves on a horizontal plane among a finite set of obstacles known as the forest of obstacles. Each k-th obstacle, $k = 1, \dots, K$, is represented by a circle with its center in point C^k and its radius r^k. Therefore, the state limitations imposed on the motion can be written as

$$\|(x(t), y(t))^T - \mathbf{C}^k\| \le r^k, \quad k = 1, \dots, K. \tag{5}$$

Note that if we have to account for the dimensions of the robot, we may assume that the robot has a circular boundary of a certain radius. This radius can be added to the radii of all obstacles (a simple case of Minkovski's sum of areas designating the obstacles and the robot), and the robot still be considered a point.

The optimal criterion can be written as

$$J = \int_0^{t^f} 1dt = t^f \to \min_{\mathbf{u} \in \mathbf{U}} \tag{6}$$

It is required to find such control function $\mathbf{u} = \mathbf{u}(t) \in \mathbf{U}$, that provides transition of the system (1) from initial state (3) into terminal state (4) within minimal transition time t^f considering state limitations (5).

Limitations (5) make the set of possible trajectories non-convex, and classical optimization methods are generally not applicable. However, for model (1), it is possible to find the optimal trajectory and optimal control function in a closed form.

It can be shown using the expressions for motion time below that for kinematic model (1), the time-optimal trajectory coincides with one of the locally shortest paths between the starting and the finishing points, and it includes in-place rotation at the start and at the finish.

III. GEOMETRY METHODS USED FOR AUTOMATIC TRAJECTORY CALCULATIONS

In this section, we discuss geometry methods used in automatic calculation of the time-optimal trajectory, as related to the kinematic model of the robot:

- Tangent search to a given circle passing through a given point;
- Bi-tangent search between two given circles;
- Filtering obstructed segments of trajectories;
- Calculating the optimal control function and the maximal speed of the robot for each segment.

A. Finding tangents

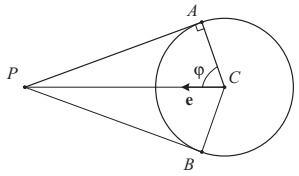

Fig. 2. Tangents from a point to a circle.

To find tangents to a given circle with its center in C and its radius r that pass through a given point P, consider fig. 2. We can find the cosine and sine of angle φ as

$$\begin{aligned} \cos\varphi &= \frac{r}{d}, \quad d = \|PC\| \\ \sin\varphi &= \sqrt{1 - \cos^2\varphi} \end{aligned} \tag{7}$$

and the tangent points A and B as

$$A = C + r\mathbf{M}^T\vec{\mathbf{e}}, \quad B = C + r\mathbf{M}\vec{\mathbf{e}}, \tag{8}$$

where M is the rotation matrix:

$$\mathbf{M} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}, \tag{9}$$

vector $\vec{\mathbf{e}} = \frac{P-C}{d}$.

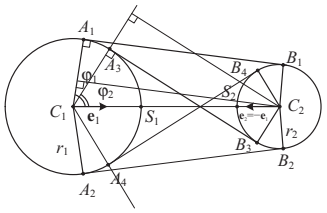

Fig. 3. Tangents between two circles.

Similarly, we can find bi-tangents between two given circles. Let the centers of the circles be at C_1 and C_2, and their radii r_1 and r_2 (fig. 3). We can find angles φ_1, φ_2 as

$$\begin{aligned} \cos\varphi_1 &= \frac{r_1 - r_2}{d}, \quad d = \|C_1C_2\| \\ \cos\varphi_2 &= \frac{r_1 + r_2}{d} \\ \sin\varphi_i &= \sqrt{1 - \cos^2\varphi_i}, \quad i = 1, 2, \end{aligned} \tag{10}$$

then construct rotation matrices

$$\mathbf{M}_i = \begin{bmatrix} \cos\varphi_i & -\sin\varphi_i \\ \sin\varphi_i & \cos\varphi_i \end{bmatrix} \tag{11}$$

and find points A_i and B_i as

$$\begin{aligned} A_1 &= C_1 + r_1\mathbf{M}_1\vec{\mathbf{e}}_1, & B_1 &= C_2 + r_2\mathbf{M}_1\vec{\mathbf{e}}_1, \\ A_2 &= C_1 + r_1\mathbf{M}_1^T\vec{\mathbf{e}}_1, & B_2 &= C_2 + r_2\mathbf{M}_1^T\vec{\mathbf{e}}_1, \\ A_3 &= C_1 + r_1\mathbf{M}_2\vec{\mathbf{e}}_1, & B_3 &= C_2 - r_2\mathbf{M}_2\vec{\mathbf{e}}_1, \\ A_4 &= C_1 + r_1\mathbf{M}_2^T\vec{\mathbf{e}}_1, & B_4 &= C_2 - r_2\mathbf{M}_2^T\vec{\mathbf{e}}_1, \end{aligned} \tag{12}$$

where $\vec{\mathbf{e}}_1 = \frac{C_2 - C_1}{d}$.

B. Detecting obstructed segments

When adding segments of a trajectory as possible candidates, we should exclude those obstructed by any obstacles. For circular obstacles, this means that we must check if a linear segment intersects with any circle.

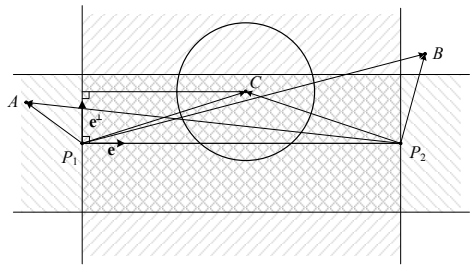

Fig. 4. Obstruction criteria. Points A and B pass, point C does not pass.

Consider a segment P_1P_2 and a circle with its center in C and its radius r (fig. 4). For the problem of moving through a forest of circular obstacles that do not intersect each other, we propose the following two criteria. First, check the distance between the center C of the circle and the line P_1P_2 and compare it to the radius of the circle r:

$$d = \|\overrightarrow{(C - P_1)} \cdot \vec{\mathbf{e}}^\perp\| < r, \tag{13}$$

where

$$\vec{\mathbf{e}} = (e_1, e_2)^T = \frac{\overrightarrow{P_2 - P_1}}{\|P_2 - P_1\|}, \quad \vec{\mathbf{e}}^\perp = (-e_2, e_1)^T \tag{14}$$

This criterion checks if the center of the circle lies within a $2r$-wide horizontal strip around the segment P_1P_2. Second, check if the center C of the circle projects onto the segment P_1P_2:

$$\left(\overrightarrow{(C - P_1)} \cdot \vec{\mathbf{e}}\right) \cdot \left(\overrightarrow{(C - P_2)} \cdot \vec{\mathbf{e}}\right) < 0, \tag{15}$$

Both dot products in (15) are positive for points that lie to the right of P_2, such as point B, and negative for points that lie to the left of point A. For points that lie inside the vertical strip between P_1 and P_2, one of the products is positive, and the other one negative.

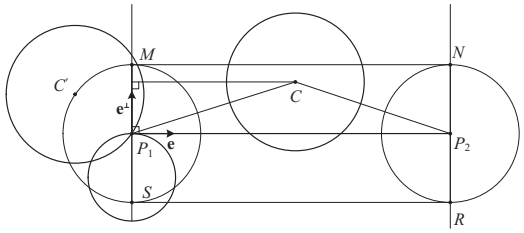

Fig. 5. Obstruction criteria. Circle with its center inside the hashed area obstructs segment P_1P_2.

In fact, in order to check if a segment is obstructed by a circle we have to check if the circle's center lies within the hashed area on fig. 5 that includes a rectangle $MNRS$ and two half-circles of radius r. However, since all linear segments are produced by tangent lines to existing circles, points P_1 and P_2 are actually tangent points of some circles and segment P_1P_2. Therefore, if C lies in one of the half-circular areas, then the circles that represent obstacles intersect each other, and this case is not considered in current article. Therefore, it is sufficient to only check if the circle center is inside the rectangular area $MNRS$, determined by the two criteria (13) and (15).

C. Maximal speed and control inputs on the segments

According to the model (1) and its control limitations (2), the maximal robot speed on the straight line segments is $v_{\max} = u_{\max}$ with control inputs at their limits:

$$u_{1,2} = u_{\max}, \tag{16}$$

and the time required to travel along a straight segment of length l is

$$t_{\min} = \frac{l}{u_{\max}} \tag{17}$$

As for circular arc segments, we can find maximal control inputs on an arc of a circle with radius r from the model definition. Note that the angular velocity of the robot is equal to the angular velocity of motion of the robot's center point O along the corresponding circle. Therefore, for counter-clockwise motion, we can write

$$\begin{aligned} v &= \frac{(u_1 + u_2)}{2} \\ \omega &= \frac{(u_1 - u_2)}{b} \\ v &= \omega r \end{aligned} \tag{18}$$

The outer (right) wheel should move with maximal velocity $u_1 = u_{\max}$, and we find from (18) that the left control input should be

$$u_2 = u_{\max}\frac{2r-b}{2r+b} \tag{19}$$

and the maximal linear speed of the robot is

$$v_{\max} = u_{\max}\frac{2r}{2r+b}. \tag{20}$$

The minimal time required to travel arc of angle φ is then

$$t_{\min} = \frac{l}{v_{\max}} = \frac{r\varphi}{v_{\max}} = \frac{r\varphi(2r+b)}{2ru_{\max}} = \frac{\varphi(2r+b)}{2u_{\max}} \tag{21}$$

When moving clockwise, the left and right control input values in (19) are swapped, but $v_{\max}$ and $t_{\min}$ remain the same.

For rotation around the center point of the robot, if the angle of that rotation is φ, then the maximum angular velocity, considering the limitations, is

$$\omega_{\max} = \frac{2u_{\max}}{b} \tag{22}$$

and the corresponding minimal time is

$$t_{\min} = \frac{\varphi b}{2u_{\max}}, \tag{23}$$

that, compared to (21), corresponds to moving along a zero-radius circle. Note that the minimal time required to move along an arc of angle φ is equal the sum of times needed to turn around by angle φ and to move at maximal speed $v = u_{\max}$ a distance of $l = r\varphi$.

D. Time-optimal path

The expression (21) can be extended to any continuous path γ as

$$t_{\min} = \frac{b}{2u_{\max}}\int\limits_{\gamma}|d\varphi| \; + \frac{l_\gamma}{u_{\max}} \tag{24}$$

This means, in particular, that any path from one point to another requires time that is the sum of integral rotation time from initial orientation to the final orientation, and the time of moving with maximum speed like in (17), where l_γ is the length of the path. The former time is constant for convex curves and is the difference between the initial and the final orientation. For non-convex paths, consider a curve with a single inflection and fixed angles at its ends (fig. 6).

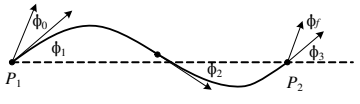

Fig. 6. Non-convex path connecting points P_1 and P_2

Since the path is continuous, if $\varphi_1 \geq 0$ and $\varphi_3 \geq 0$, then at the inflection $\varphi 2 \leq 0$. The integral angle is the sum $|\varphi_1 - \varphi_0| + |\varphi_2 - \varphi_1| + |\varphi_3 - \varphi_2| + |\varphi^f - \varphi_3|$, and it is minimal when $\varphi_{1,2,3} = 0$, that can be achieved when the curve is a straight segment. Similar reasoning is applicable to curves with more than one inflection. So, in any case, the straight motion with rotations in the ends is time-optimal.

When moving from one point to another with one obstacle in the way, from expression (24) it follows that the shortest path is time-optimal. This is as much as can be said, however, as when there are more obstacles, we must consider the slowdown on the arc segments of the path. Still the minimum of travel time is delivered by a path that consists of straight segments and arc segments, and therefore is one of the local minima in the shortest path problem.

When the optimal motion consists of interleaving straight segments and arc segments, it is provided by piecewise constant control inputs (16) and (19), and the discontinuities occur at the connection points of these segments.

IV. TRAJECTORY SEARCH ON A GRAPH

A. Building a graph

The time-optimal trajectory of the robot is one of the locally shortest paths between the two given points, with the addition of two rotations at the starting point and at the finishing point. For the circular obstacles, in particular, the time-optimal path consists of interleaving straight segments and arc segments. The segments connect in points on the circles that are tangent points between the lines and the obstacles. We shall build a graph of possible trajectories where the edges are the segments, and the vertices are their connection points, and also the starting point and the finishing point. The graph is a directed one, meaning that a connection from vertex V to vertex W does not imply a reverse connection from W to V.

All points that designate vertices of the graph have three values: the x and y coordinates of each point on the plane, and θ – the direction of the robot when it passes the point. We shall call it a *directed point*.

Three types of connections are possible: a *line* connection, an *arc* connection and a *rotation* connection. For rotation, the cartesian coordinates of its beginning and end are the same, and the θ-value shows the initial and terminal heading of the robot.

Let the starting point be denoted as S, the finishing point as F. When we say that we add a connection between any two points, we mean that this connection is added only if the corresponding line segment is not obstructed by any of the obstacles. For arc connections, since we only consider non-overlapping obstacles, no obstruction is possible, and no checks are done.

We first add points S (start) and F (finish) to the graph and check if the straight motion from the start to the finish is possible. If so, then no other search is needed, because it is the absolute minimum. We still have to add, however, vertices that correspond to the directed point S_F located at the start and directed towards the finish, and F_S, located at the finish and directed opposite of the start, and rotation connections between S and S_F and between F_S and F.

Rotation connections have zero length, but the time required to turn around a point is evaluated as in (23). Rotation is possible in one of the two directions, and its time is less if the angle of rotation is less.

Connections between the start and i-th circle are straight segments of tangents that touch the circle at points A_{iSR} and A_{iSL}. In the subscript, i designates the number of the circle in the list of all circular obstacles, S stands for start, and R and L denote the right and left side of the circle when we look from that circle to the starting point. Similarly, we define A_{iFR} and A_{iFL}. We add the corresponding vertices to the graph, and we also have to add vertices S_{iR} and S_{iL} that designate ends of rotations towards A_{iSR} and A_{iSL}, respectively, and the same is done for the finish, except that the direction is taken towards the finishing point. Line connections are then added between S_{iR} and A_{iSR}, S_{iL} and A_{iSL}, A_{iFR} and F_{iR}, and A_{iFL} and F_{iL}, and rotation connections from vertex S to all vertices S_{i*} and from all F_{i*} to F, the asterisk denoting arbitrary available index.

Connections between circles are added as follows. Each pair of circles with numbers i and j adds four tangent points on circle i and four points on circle j. We shall denote the points as A_{ijRR}, A_{ijRL}, A_{ijLR}, and A_{ijLL}, where i indicates the circle where the point is located, j – the circle towards which the line segment is directed, first literal index R or L denotes the side of circle i if we look at circle j, and the second literal index – the side of circle j towards which the segment is directed, looking from circle i. If we refer to 3, for example, $i = 1$, $j = 2$, and $A_{12LL} = A_1$, $A_{12RR} = A_2$, $A_{12LR} = A_3$, $A_{12RL} = A_4$, $A_{21LL} = B_2$, $A_{12RR} = B_1$, $A_{12LR} = B_3$, $A_{12RL} = B_4$. The corresponding line connections are added if the segments are not obstructed.

Finally, for each circle, we add arc connections for all points A_{i***} that belong to that circle i. To decrease the number of edges in the graph, we do the following check.

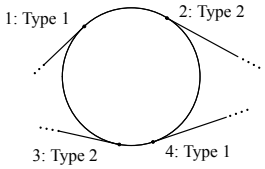

Fig. 7. Connection types: tangent approaching from the left (type 1) and from the right (type 2).

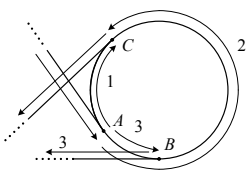

Fig. 8. Connection types: tangent approaching from the left (type 1) and from the right (type 2).

The tangent points on a circle can be divided into two types (fig. 7). At type 1 points, the tangent segment enters from the left if we look at that point from the center. At type 2 points, the tangent segment enters from the right. Note that the optimal trajectory can only enter at type 1 point, continue towards the direction determined by that segment, and exit from type 2 point (or vice versa), so trajectory that enters from upper-left at point A and exits at point B towards direction 3 cannot be optimal, and should not be included in the search. Besides, if the angle of the arc between two points is greater than π, then the cusping path that includes points A and C (8, path 1) is shorter than the smooth one (8, path 2), and the smooth segment can be excluded from search as being a priori non-optimal. However, the cusping path cannot be optimal as well. Thus all arc connections that have angles greater that π can be skipped at the connection adding stage.

In proposed indexation, all points that have literal R in the third position are of type 1, and all point that have literal L are of type 2. Therefore, when adding connections, we connect all "L" points with all "R" points by counter-clockwise arcs though checking if the arc angle does not exceed π, and vice versa.

B. Finding optimal path on a graph

The constructed graph consists of all points that connect segments of possible optimal trajectories as its vertices, and its edges are the connections between these points that may be segments or the optimal trajectory. Since we can evaluate both the length and the travel time of all segments, we can find the shortest path (or the fastest path), we use Dijkstra's algorithm [12]. An A* algorithm [13] can also be used, it may prove faster.

The Dijkstra's algorithm can be summarized as follows. All vertices except the starting one are assigned their path lengths as $+\infty$. The starting vertex is assigned zero length. The term "length" can mean travel time or actual distance. The vertices are marked as "visited" or "unvisited", and at the start all vertices are marked as "unvisited". At each step, an "unvisited" vertex v_i is chosen that has minimal length, and for all neighboring vertices v_j the path length is calculated, adding edge length between v_i and v_j to the saved length of v_i. If at some moment the newly calculated path length of v_j is less than the saved one, the new length is saved to this vertex v_j. At this time, the predecessor of vertex v_j is assigned to v_i. We shall also prefer trajectories that contain fewer segments, so we calculate the number of hops between points, and keep those trajectories with equal length that have fewer hops. A vertex is marked as "visited" when all its outgoing neighbors are checked this way. When the dedicated (finish) vertex becomes "visited", the algorithm breaks. If after the break, the target vertex has infinite length, this means that the graph is disjunct, and no path exist between the start and the finish.

The path towards the finish can be then found moving in reverse from the finish to the start by examining the saved predecessors of the vertices.

V. COMPUTATIONAL EXPERIMENT

A Python program was implemented to illustrate the method of tangent search. The program takes as its input a scenario file that includes parameters of the circular obstacles, the starting and finishing point, the geometrical parameter b and the control limitation $u_{\max}$. We can choose to search for the fastest or the shortest path.

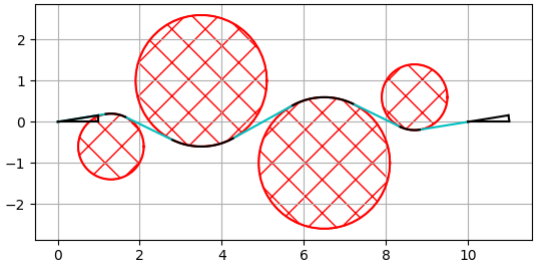

Fig. 9. Optimal trajectory from point $(0, 0, 0)$ to point $(10, 0, 0)$ with 4 obstacles.

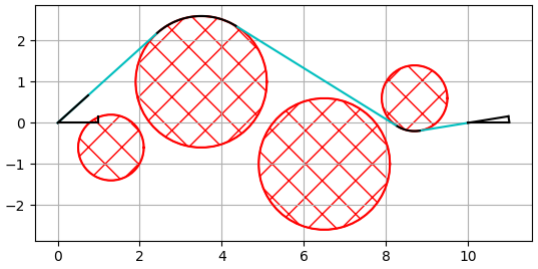

Fig. 10. Optimal trajectory from point $(0, 0, 0)$ to point $(10, 0, 0)$ with 4 obstacles with higher value of $b = 10$.

An example of optimal trajectory search is presented at fig. 9. The starting point is taken as

$$(0, 0, 0)$$

where the third component is the initial orientation of the robot. The finishing point is $(10, 0, 0)$. The values of $u_{\max}$ and b are taken as 1. The coordinates and radii of the circles are:

1) $x = 3.5,\ y = 1.0,\ r = 1.6$
2) $x = 6.5,\ y = -1.0,\ r = 1.6$
3) $x = 1.3,\ y = -0.6,\ r = 0.8$
4) $x = 8.7,\ y = 0.6,\ r = 0.8$

At the drawing, the arcs are shown in black and the straight segments are shown in cyan. The rotations are depicted as angles, with their sides of unit length, and the starting side marked with a short tick. The path length from the start to the finish is 10.64 units, while the time required to travel along the trajectory is 12.36 units. Higher values of b result in the robot slowing down more on the arc segments. So, with the value of $b = 10$, the velocity reduction results in selecting another locally shortest path around the obstacles (fig. 10) with travel time of 22.09 units. The previously fastest path now evaluates to travel time of 22.9 units, and is slower than the new minimum.

VI. CONCLUSIONS

In the article, we have shown that the time-optimal trajectory for kinematic motion model of a differential wheeled robot is the locally shortest path between two given points – the start and the finish, that consists of interleaving straight segments and arc segments. Expressions to calculate the travel time along this path are provided. The expressions for control inputs required to move the robot along the optimal path segments are given in section III,C. They define piecewise-constant inputs at each segment.

The kinematic model of a wheeled robot is often used as a benchmark model for testing numerical methods of finding optimal control functions, and time-optimal problem is a common problem investigated in robot motion. Having a tool that solves a time-optimal problem is convenient for validating these numerical methods.

Besides, the finite search even on large sets of obstacles is fast and can be done in real time, so the result provided by the discussed method can be used as an initial approximation even when more complex models are considered.

REFERENCES

[1] L.S. Pontryagin. "Maximum principle in optimal control". Moscow, Nauka, 1989, 79 p. (in Russ.)

[2] N.I. Grachev, Yu.G. Evtushenko. "Biblioteka programm dlya resheniya zadach optimal'nogo upravleniya" [A collection of programs for solving optimal control problems]. Computational Mathematics and Mathematical Physics, Vol. 19, No. 2, pp. 367-387, 1979. (in Russ.)

[3] R.P. Fedorenko. "Priblizhennoe resheniye zadach optimal'nogo upravleniya" [Approximate solving of optimal control problems]. Moscow, Nauka, 1978. 488 p. (in Russ.)

[4] L.E. Kavraki, P. Svestka, J.C. Latombe, M.H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, Vol. 12(4), pp. 566-580, 1996.

[5] S.M. LaValle. "Rapidly-exploring random trees: A new tool for path planning". Technical Report TR98-11. Department of Computer Science, Iowa State University, 1998.

[6] J. Nocedal, S.J. Wright. "Numerical Optimization", 2nd ed, Springer, 2006.

[7] R.S. Sutton, A.G. Barto. "Reinforcement Learning: An Introduction", MIT Press, 2018.

[8] A. I. Diveev, and S.V. Konstantinov. "Study of the practical convergence of evolutionary algorithms for the optimal program control of a wheeled robot". Journal of Computer and Systems Sciences International, Vol. 57, pp 561-580, 2018.

[9] A.I. Diveev, and N.C. Mendez-Florez. "Synthesis of a spatial stabilization for a mobile robot by means of symbolic regression learning", Vestnik RUDN, Series: Engineering research, Vol.22, No.2, pp. 129-138, 2021. (in Russ.)

[10] V.A. Bereznev. "The principle of dividing feasible trajectories in a robot control problem", Procedia Computer Science, P.456-459, 2021.

[11] A.N. Daryina, A.I. Diveev, D.Y. Karamzin, F.L. Pereira, E.A. Sofronova, and R.A. Chertovskikh. "Regular Approximations of the Fastest Motion of Mobile Robot under Bounded State Variables", Computational Mathematics and Mathematical Physics, Vol. 62(9), pp. 1539-1558, 2022.

[12] E.W. Dijkstra. "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, 1959.

[13] P.E. Hart P.E, N.J. Nilsson N.J., and B. Raphael. "A formal basis for the heuristic determination of minimum cost paths:, IEEE Transactions on Systems Science and Cybernetics, Vol. 4(2), pp. 100-107, 1968.