DOI 10.2478/ama-2025-0020

3K METHOD: TIME-OPTIMAL PATH PLANNING FOR FIELD ROBOT

Konrad K. KWAŚNIEWSKI*, Zdzisław GOSIEWSKI

* Faculty of Electrical Engineering, Bialystok University of Technology, ul. Wiejska 45 C, 15-351 Bialystok, Poland

kwasniewski.konrad.krzysztof@gmail.com, gosiewski@post.pl

received 10 October 2023, revised 31 May 2024, accepted 25 July 2024

Abstract: In this study, a hybrid genetic-geometrical path finding method is presented. Its main feature is the division of the path-finding process into global and local path-finding to achieve a trajectory optimized under the shortest travel time condition in an environment filled with obstacles. To improve the reliability of the algorithm, a safety zone around obstacles is included. In this zone, the maximum velocity allowed for a robot is additionally limited to decrease the probability of collision due to noise in obstacle mapping, distraction from terrain irregularities or malfunction of the steering system. The simulation and real world experiment results are presented in another paper.

Key words: mobile robot, path planning, path optimization, robot rover, autonomous, genetic algorithm

1. INTRODUCTION

Nowadays, the usage of robots in production services is constantly increasing. This trend extends even to domestic applications, such as vacuum cleaners and lawnmowers. The expanding range of fields where autonomous robots can be employed presents us with new challenges. Widely known are autonomous cars, which are specifically adapted to road environments. However, off-road areas are much more prevalent on Earth. Tasks such as rescue actions, emergency transportation, patrol, and exploration in natural disaster areas often occur in regions with limited or nonexistent transportation infrastructure. Therefore, robust navigation methods that consider travel time constraints are crucial for further development in these types of terrains. Path planning and obstacle avoidance methods are often based on graph algorithms such as A* [22, 1], Dijkstra [14], or D* [20]. The path obtained by those methods has to be smoothed before applying it to a controller [5]. Another approach is to use artificial neural networks (ANN) [3, 19, 21, 25]. An alternative way is to use optimization methods. Biology-inspired genetic algorithms [10, 7, 9, 14], ant colony optimization [22], particle swarm optimization [24], chicken swarm optimization [6], cuckoo search optimization [2], grey wolf algorithm [16], whale optimization algorithm [17]. Other noteworthy approaches are potential fields [11] and fuzzy-based potential fields [4]. An interesting field of path planning are methods based on the Dubins path. Worth mentioning are especially methods using the coverage path planning approach, which can be used for both a priori known [12] and unknown [13] environments.

2. 3K METHOD OVERVIEW

When it comes to avoiding an obstacle, the solution is trivial. The robot can ride on its left or right side. The problem becomes more complex when there are multiple obstacles. But if we want to go from point A to point B as fast as possible, the task becomes very complex. When a robot knows only about obstacles in its surroundings it does not have enough knowledge to choose the best path in the general context. On the other hand, if it follows a path, that was found using only previously mapped obstacles, it cannot omit a new object in the working area. The 3K method combines two strategies: it finds a global path using a priori known obstacles from maps and optimizes it under the travel time criterion; then, during the ride, it uses the global path as a set of waypoints and corrects the trajectory by finding a local obstacle-free path. These local obstacles are detected autonomously by the robot. The local path is also selected under the shortest travel time condition.

The characteristic features of the 3K method in comparison to the other ones are obstacles modeled as circles, division of the problem to the local and the global scales and introduction of the velocity reduction area around obstacles for more precise navigation in the immediate vicinity of obstacles, especially those not known a priori, for safety reasons.

The main algorithm is presented in Fig. 1. The current robot position is considered a start position, while the ordered travel point is considered a destination point. At first, the map of a priori known obstacles is loaded. Then the global path is found, and the travel begins. The robot moves along the sequence of the global path waypoints. If at least one obstacle is detected, the local path between the current robot position and one of the more distant global waypoints is determined, and the robot starts following the local path. The local path computation is repeated as long as at least one local obstacle is in the robot's view range. When the robot has avoided all the obstacles, it continues following the global waypoints. The 3K method ends when the last global waypoint, the destination point, is reached.

2.1. Map and obstacle models

The map is modeled in a 2D Cartesian coordinate system. This simplification, in relation to the real form of Earth, facilitates

its mathematical description and is a good approximation on short distances on the planet surface. The map is stored as a list of obstacles. It is shown in Fig. 2.

All obstacles are modeled as circles. In the case of bigger obstacles or those of complex shape, they can be either circumscribed by a large circle or covered by multiple smaller ones. Each i-th circle O_i is described by two values: radius $O_{r,i}$ [m] and center point $O_{C,i}$ [m, m] where $O_{C,i} \in C$. The axis of the real part is identical to the X axis of the robot-centered coordinate system and, similarly, the imaginary part to the Y axis of the robot-centered coordinate system. The advantage of this model is its mathematical simplicity, which allows for easy collision detection.

Around each obstacle, the velocity reduction area is applied. It is the additional circle, in which the velocity of the robot has to be reduced to provide better precision of steering when the robot moves close to the obstacle. This approach reduces the probability of a collision in the case of an error in the obstacle position estimation.

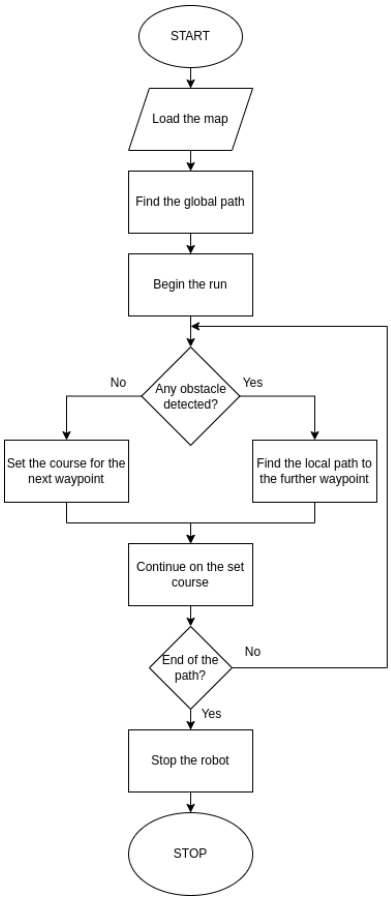
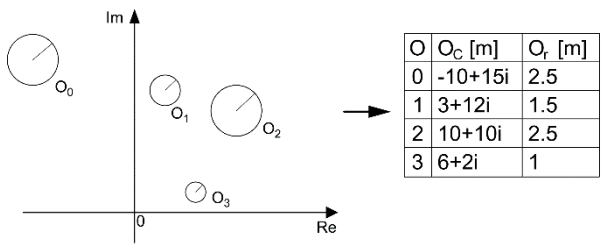


Fig. 1. Flowchart of 3K method



O	O_C [m]	O_r [m]
0	-10+15i	2.5
1	3+12i	1.5
2	10+10i	2.5
3	6+2i	1

Fig. 2. Example of map model

2.2. Path model

The path of the global path method is modeled using a B-spline curve, which is a special case of a NURBS curve (Non-Uniform Rational B-Spline) [18]. The B-spline has uniformly distributed knots, unlike the NURBS, in which the knots can be distributed non-uniformly. The shape of the B-spline is defined by a vector of control points and a degree of the curve. The number of control points has to be at least greater by one than the curve degree. In curves of a higher degree, more control points are considered when particular points of the curve are computed. The main advantage of the B-spline is its property, that translation of a control point of a curve modifies it only to a limited distance from the control point (depending on the curve degree). The size of the shape modification depends on the distance of the translation. It makes the B-spline very well suited to be a model of a path that undergoes operations of an optimization algorithm.

2.3. Collision detection

A collision between the B-spline and a circle can be detected by computing a set of equidistant curve points S and testing, if any of them lies inside the currently considered circle (Fig. 3). In the case of obstacle avoidance, however, where an optimization algorithm is used, the binary information is not enough. A smoother solution is required. Hence, the collision depth is adopted instead. It is computed as follows. Firstly, the closest point p_c of the S set to the obstacle center O_C is selected. Then the collision depth is the difference between the obstacle radius O_r and the distance between the obstacle center O_C and the p_c point, divided by the radius O_r. Then the collision depth value is always in the range [0; 1]. It is shown in Fig. 4 and given by the formula (1).

$$m_k = \frac{O_r - |O_C p_C|}{O_r}, |O_C p_C| \leq O_r, O_r \neq 0 \quad (1)$$

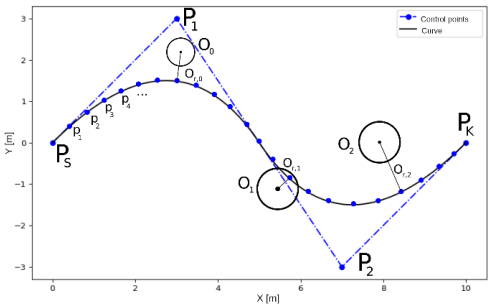


Fig. 3. Example of collision detection between B-spline and circle-shaped obstacles

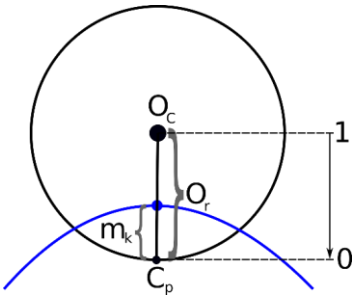


Fig. 4. Visualization of the collision depth

DOI 10.2478/ama-2025-0020

3. VELOCITY PROFILER

To find a path of the shortest travel time, it is necessary to know the path travel time itself. To estimate it, the velocity profile of a path is needed. The travel, however, has to fulfill certain constraints:

- the robot must not exceed the given maximum velocity,
- the robot must not exceed the maximum velocity of considered segment of the path, to secure the robot against skidding,
- the acceleration must not exceed the maximal acceleration value, either set or achievable by the robot,
- the deceleration, as in the case of the acceleration, must not exceed the maximal acceleration value, either set or achievable by the robot.

The acceleration and deceleration rate in the 3K method are considered as constant in time. The difference between the constant rates and the real ones is negligible because of many other factors present in a real environment, which can greatly affect the robot during the journey.

The computation of the velocity profile consists of the following steps:

1. computation of the maximum permissible velocities on all the segments of the path (between waypoints) – the maximum velocities profile,
2. reduction of velocities from the maximum velocities profile in such a way that the given acceleration value is maintained,
3. reduction of velocities from the reduced velocities profile in such a way that the given deceleration value is maintained,
4. computation of the travel time of the individual path segments and the total travel time.

The main factor limiting the robot's velocity is the friction between the robot wheels and the ground. The speed must be low enough to prevent the robot from getting out of a track due to the centrifugal force. The formula for the maximum velocity v_{max} on the particular path segment (4) is derived from the formulae for the friction T (2) and the centrifugal F_c (3) forces. The maximum velocity is a velocity of the state where both forces are equal, i.e. $T = F_c$.

$$T = \mu N = \mu m g \tag{2}$$

$$F_c = \frac{m v^2}{r} \tag{3}$$

$$v_{max} = \sqrt{\mu g r} \tag{4}$$

where m is the mass of the robot [kg], g is standard gravity acceleration [m/s^2], v - velocity [m/s], r - radius of the path segment [m]. In the case of a straight segment the maximum velocity is infinite, which is in fact the maximum velocity allowed for the robot to reach.

In the velocity reduction steps, the new velocity for the $i+1$-th waypoint is computed from the formula (5). The start and the end point velocities are set at the beginning of the algorithm and are equal to 0. They remain intact during the whole process.

$$v_{i+1} = \sqrt{2 a d_i + v_i^2}, i \in [0, n-2], a \leq 0 \tag{5}$$

where a - a set acceleration or deceleration [m/s^2], d_i - Euclidean distance between i-th and $i+1$-th waypoint, n - number of waypoints. This formula is derived from the equation (6) [8].

$$v^2 = v_0^2 + 2a(x - x_0) \tag{6}$$

The acceleration and deceleration steps are computed in the same manner, although as the deceleration is a reverse operation to the acceleration, the reduction process has to proceed from the last waypoint to the first one.

The partial travel times, which mean times between each pair of the following waypoints, is computed by formula (8) which is derived from (7) [8].

$$(x - x_0) = \frac{1}{2}(v_0 + v)t \tag{7}$$

$$t_i = \frac{2 d_i}{v_i + v_{i+1}}, i \in [0, n-2] \tag{8}$$

where in (7): x - current distance, x_0 - initial distance, v_0 - initial velocity, v - current velocity, t - travel time; in (8): t_i - travel time between i-th and $i+1$-th waypoints, d_i - distance between i-th and $i+1$-th waypoints, v_i velocity at i-th waypoint, v_{i+1} velocity at $i+1$-th waypoint, n – number of waypoints.

Total travel time t_c [s] is a sum of all the partial times (9).

$$t_c = \sum_{i=0}^{n-2} t_i \tag{9}$$

4. GLOBAL PATH METHOD

The global path method is based on a genetic algorithm (GA). It is used for finding a collision free path, whose shape is optimized to minimize the travel time. The features of the used GA are:

- roulette parents selection operator,
- arithmetical non-symmetric crossing operator,
- population size of 100 individuals,
- elitism: the best individual of a current generation is transferred into a next one,
- mutation operator of low probability and a narrow range of a gene value change,
- a chromosome is a sequence of following B-spline control points,
- control points are defined as complex numbers,
- optimization takes place under two criteria: collision (obstacle avoidance) and travel time.

The collision criterion is measured with a collision depth value which is introduced by the formula (1). It is a sum of the collision depths between the considered path and all the obstacles. The travel time is computed for the considered path by the velocity profiler. Sets of the fitness values, that mean the collision and the travel time, of the all individuals are then remapped to a new domain [0; 1] separately. Then they are weighted and summed. It is shown by the equation (10).

$$F = M_k c_m + T c_p; c_m + c_p = 1; c_m, c_p > 0 \tag{10}$$

where F is a vector of the fitness values of the whole population, M_k is a vector of the remapped collision criteria, T is a vector of the remapped travel times, c_m and c_p are coefficients of the collision criterion and the travel time respectively.

The initial population is generated by random modifications of the curve, that is built of equidistant control points, which are located on the straight line between the first and the last control point. The first point is equal to the robot start position and the last point is equal to the destination point. The robot start position and the destination point are converted from the ECEF coordinates system to the local coordinates system of the robot. The initial

straight line variant of the curve is always included in the initial population. The modification range is limited to the area around known obstacles. The algorithm stops when the differences between the individuals become smaller than a set threshold.

When the algorithm stops, the result path is returned, that means the path with the lowest fitness value in the last generation of the individuals. Before use, it is converted to a set of the following waypoints.

5. LOCAL PATH METHOD

The local path is computed in a geometrical way. The general idea of the method is presented in the Fig. 5. The map and obstacle models are the same as described in section 2. The method looks for a path constructed with two straight line segments between the local start P_S and the local destination P_K points. At first, rays, $l_{s,i}$ from P_S and $l_{k,i}$ from P_K , tangent to each obstacle are computed. It gives two pairs of the rays for each obstacle. To this set, a straight path between P_S and P_K points is added. Next, the intersection points D_i between the rays are calculated. This process is shown in the Fig. 6. The paths are created from segments of intersecting rays from the points P_S and P_K , which ends at their intersection point. Each path is then tested for a collision. If it is detected for at least one obstacle, the path is removed from the set.

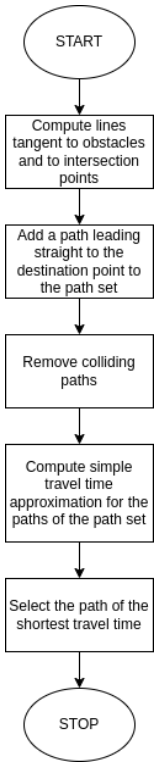


Fig. 5. Flowchart of the local path finding method

The same process is done for the velocity reduction areas instead of the obstacles. Collisions, however, still are computed for the obstacles.

For the remaining paths, approximate travel times are computed. The approximate travel time t_j for the j-th path is described by the formula (11).

$$t_j = \frac{l_j - c_j}{v_{max}} + \frac{c_j}{v_{min}} \tag{11}$$

where l_j is the length of the j-th path, c_j is the sum of the chord lengths in the velocity reduction areas i.e. these parts of the path, that lie in the circles of the velocity reduction areas, v_{max} is the maximum allowed velocity for the robot, v_{min} is the maximum allowed velocity in the velocity reduction areas.

When the approximated travel times of every path are known, then the path of the shortest time is selected from the set as a result path and the local path algorithm ends.

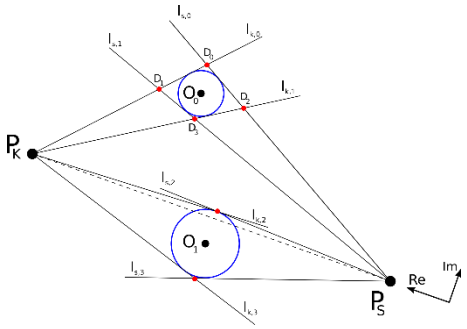


Fig. 6. Visualization of local path computation

6. ROBOT NAVIGATION METHOD

The robot navigation method joins the results of the global path method (sec. 4) and the local path method (sec. 5). It consists of three layers: movement along the global path, deviation from the global path to avoid local obstacles (movement along the local path), return to the global path.

6.1. Movement along the global path

The global path is set for the algorithm as a sequence of the waypoints. The robot sets the course to the following waypoints. The current waypoint is changed to the next one, when the distance between the current waypoint and the robot is shorter than the given one. This distance is named point change distance. It is often set up to several meters. A shorter distance forces the robot to follow the path more strictly, a longer one is less prone to errors of losing the path.

Before beginning, the velocity profile for the global path is computed. During the ride, the robot velocity is set to the value from the profile that corresponds to the currently selected waypoint.

6.2. Movement along the local path

If at least one obstacle is detected, then the local path method is used to create a path. The local destination point is set to a distant point from the set of not yet reached waypoints. The minimal distance for the selection is set by the robot operator. The result is a navigation point, which the robot sets the course to. The algorithm is frequently repeated while any obstacles are in sight. The robot velocity is set dependently on the local path - if it goes through the velocity reduction area (v_{max} (11)) or not (v_{min} (11)).

DOI 10.2478/ama-2025-0020

6.3. Return to the global path

When again no obstacle is detected, the closest global waypoint to the robot is set as the navigation point. The point change distance smooths the return to the global path, because of the navigation point changing to the next waypoints as the robot comes closer to them.

6.4. Termination

The algorithm stops along with the robot, when the global destination point is reached within the given accuracy.

7. SUMMARY

The 3K method described in the paper divides the path planning problem into two cases: the global context and the local context. Two distinct methods to both global and local path finding were developed in order to achieve necessary performance for successfully controlling the robot.

The global method is based on a genetic algorithm, and it uses the map of a priori known obstacles to produce a path. The local method is based on the geometrical path planning approach with numerous simplifications of the problem model. It uses the data about the obstacles that are detected during the run.

The third method was designed to control the robot using data obtained from the global and the local path finding methods. It leads the robot along the global path, but when an unknown obstacle is found, the local path is used to avoid it.

The main purpose of the 3K method is to minimize travel time of the robot in partially unknown environments. The results of the experiments in simulations and in the real world are described in a following paper.

REFERENCES

1. Alazzam H, AbuAlghanam O, Sharieh A. Best path in mountain environment based on parallel A* algorithm and apache spark. The Journal of Supercomputing. 2022; 1–20.
2. Alymani M, Alsolai H, Maashi M, Alhebri A, Alshahrani H, Al-Wesabi FN, Mohamed A, Hamza MA. Dispersal foraging strategy with cuckoo search optimization based path planning in unmanned aerial vehicle networks. IEEE Access 11. 2023; 31365–31372.
3. Bozek P, Karavaev YL, Ardentov AA, Yefremov KS. Neural network control of a wheeled mobile robot based on optimal trajectories. International journal of advanced robotic systems 2020;17: 2.
4. Cao X, Zuo F. A fuzzy-based potential field hierarchical reinforcement learning approach for target hunting by multi-auv in 3-d underwater environments. International Journal of Control 2021;94(5):1334–1343.
5. Duan S, Wang Q, Han X. Improved a-star algorithm for safety insured optimal path with smoothed corner turns. Journal of Mechanical Engineering. 2020;56(18): 205–215.
6. Fu W, Wang B, Li X, Liu L, Wang Y. Ascent trajectory optimization for hypersonic vehicle based on improved chicken swarm optimization. IEEE Access 7. 2019;151836–151850.
7. Gosiewski Z, Kwaśniewski K. Time minimization of rescue action realized by an autonomous vehicle. Electronics 9. 2020;12: 2099.
8. Halliday D, Resnick R, Walker J. Fundamentals of physics. John Wiley & Sons; 2013.
9. Jiang A, Yao X, Zhou J. Research on path planning of real-time obstacle avoidance of mechanical arm based on genetic algorithm. The Journal of Engineering. 2018;16: 1579–1586.
10. Kwaśniewski KK, Gosiewski Z. Genetic algorithm for mobile robot route planning with obstacle avoidance. acta mechanica et automatica. 2018; 12(2): 151–159.
11. Li G, Yamashita A, Asama H, Tamura Y. An efficient improved artificial potential field based regression search method for robot path planning. In 2012 IEEE International Conference on Mechatronics and Automation. 2012; 1227–1232.
12. Li L, Shi D, Jin S, Yang S, Zhou C, Lian Y, Liu H. Exact and Heuristic Multi-Robot Dubins Coverage Path Planning for Known Environments. Sensors. 2023;23(5):2560.
13. Li L, Shi D, Jin S, Yang S, Lian Y, Liu H. SP2E: Online spiral coverage with proactive prevention extremum for unknown environments. Journal of Intelligent & Robotic Systems; 2023; 108(2): 30.
14. Lo CC, Yu SW. A two-phased evolutionary approach for intelligent task assignment & scheduling. In 2015 11th international conference on natural computation (ICNC). 2015; 1092–1097.
15. Norhafezah K, Nurfadzliana A, Megawati O. Simulation of municipal solid waste route optimization by dijkstra's algorithm. Journal of Fundamental and Applied Sciences 9. 2017; 732–747.
16. Pawlowski A, Romaniuk S, Kulesza Z, Petrović M. Trajectory optimization using learning from demonstration with meta-heuristic grey wolf algorithm. IAES International Journal of Robotics and Automation (IJRA). 2022; 11(4): 263–277.
17. Petrović M, Miljković Z, Jokić A. A novel methodology for optimal single mobile robot scheduling using whale optimization algorithm. Applied Soft Computing 81. 2019;105520.
18. Piegl L,Tiller W. The NURBS book. Springer Science & Business Media; 1996.
19. Salt L, Howard D, Indiveri G, Sandamirskaya Y. Parameter optimization and learning in a spiking neural network for uav obstacle avoidance targeting neuromorphic processors. IEEE transactions on neural networks and learning systems. 2019; 31(9): 3305–3318.
20. Shi J, Liu C, Xi H. Improved d* path planning algorithm based on CA model. Journal of Electronic Measurement & Instrumentation; 2016.
21. Singla A, Padakandla S, Bhatnagar S. Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge. IEEE Transactions on Intelligent Transportation Systems. 2019; 22(1): 107–118.
22. Wang Z, Zeng G, Huang B, Fang Z. Global optimal path planning for robots with improved A* algorithm. Journal of Computer Applications. 2019;39(9): 2517.
23. Yi Z, Yanan Z, Xiangde L. Path planning of multiple industrial mobile robots based on ant colony algorithm. In 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing. 2019; 406–409.
24. Zhang T, Xu J, Wu B. Hybrid path planning model for multiple robots considering obstacle avoidance. IEEE Access 10. 2022;71914-71935.
25. Zhu H, Ouyang H, Xi H. Neural network-based time optimal trajectory planning method for rotary cranes with obstacle avoidance. Mechanical Systems and Signal Processing 185. 2023;109777.

Konrad K. Kwaśniewski: https://orcid.org/0000-0003-2528-8904

Zdzisław Gosiewski: https://orcid.org/0000-0002-7437-2574