Dubin's Curve of RRT* Merged With A*

Heru Suwoyo
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Fahrudin
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Abstract: One of the fundamental problems in mobile robotics is a robot path planning of the mobile robot through its environment. Path planning problem for the mobile robot with differential constraints using modified RRT (Rapidly exploring random tree) algorithm based on Dubin's curves. the planning problem is considered as a problem of finding a feasible path between the initial and goal point in a static environment with obstacles. This process can be conducted either using local information from sensors or by emloying global a-priori known information about robot's environment. The problem is how to generate a path from the beginning to the destination point gradually during movement using modified RRT (Rapidly exploring random tree) algorithm based on Dubin's curves. Combining the dubin curve RRT* algorithm with A* is a new method that can calculate the entire path from the starting point of the destination before moving using global information about the map. The purpose of making the path is to make it easier for the operator to determine the path that must be traversed by the robot. The way the robot works is to read the line made by the operator using matlab based on the map, then matlab will calculate the distance of the path to be traversed using an algorithm from the star point to the goal point. Based on the simulation results that have been carried out this method is more efficient when compared to the RRT* or A* algorithms. because this algorithm can produce a path with the shortest path with a fast time to get to the destination point without crashing into obstacles. By adding a new algorithm to find a new path optimally to get a path that is close to optimal by combining and adjusting several feasible paths and also adding a searching-based algorithm, namely A* combined with Dubin Curve- RRT* is sampling based, where the A* algorithm has a function heuristic used to increase the cost and time of searching.

Keywords: Algorithm, curva dubin, RRT*, A*

Correspondents Author:
Heru Suwoyo, Department of Electrical Engineering, Universitas Mercu Buana, Indonesia
Email : heru.suwoyo@mercubuana.ac.id

Introduction

An alternative approach to the problem with reversals was proposed by Soueres and Laumond (Jacobs et al., 1991). They tie the Pontryagin's optimality principle with geometric reasoning and arrive at the optimal solution via partitioning of C-space into regions with uniform properties of path optimality. In the context of robotics, the original Dubins problem of constructing a smooth path has a significance of its own (Daniel Tenezaca et al., 2020). In many motion planning tasks, such as in the aircraft control, motion reversals are not feasible. Or, if the shortest time path, rather than the shortest path, is desired, the solution is likely to be a smooth path, because the deceleration, stop, and acceleration at the reversal cusps add time to the path execution. Unfortunately, Dubins' problem with smooth paths is not a subset of the Reeds Shepp's problem the sufficient set of the former is not contained in the sufficient set of the latter. Also, the techniques proposed in the techniques proposed in are not directly applicable to the smooth path case (Soueres & Laumond, 1996).

To use Dubins's result for the shortest path calculation, one would need to explicitly calculate the lengths of all arcs and straight-line segments in the Dubins set, and then choose the shortest of the computed paths (Tenezaca et al., 2019). The time necessary for this calculation may become a bottleneck in time-constrained applications, as e.g. in real- time robot motion planning—which is one motivation for this work. Another motivation comes from problems where one looks for the shortest path from a point to a manifold in the C-space. For example, in sensor-based obstacle avoidance, when planning an arrival to some intermediate point P on the obstacle boundary, the current sensing data may suggest that in order not to collide with the obstacle, the orientation angle β at P must be within some sector of angles (which may include, e.g. the tangent to the obstacle at P).

Finding the shortest path to P under this constraint corresponds to finding the shortest path to a line in C- space. In this work, we propose a scheme which allows one to select the shortest path from the Dubins set D directly, without the usual exhaustive calculation of its elements. The scheme is based on a rather suggestive fact, that the elements of the Dubins set can be classified into a small number of the so-called equivalency groups, based on the angle quadrants of the corresponding pairs of the initial and final orientation angles. Each equivalency group consists of a few classes of paths, such that any path in a group is equivalent, up to an orthogonal transformation, to any other path in the same group. This means that the optimal path analysis can be reduced to fewer terms. Further, a simple logical classification of the equivalency groups can be built which points directly to the optimal path (Tenezaca et al., 2019).

The analysis necessary for solving our classification problem turns out to become simpler if it is divided into two cases, which can be called the long path case and the short path case. Our approach is equally applicable to both cases, with minor differences between the resulting computational schemes. For the sake of example, we consider here only one case, the long path case, which seems to be of more interest from the standpoint of applications and the computational savings (Račinský, 2016). More precisely, the "long paths" are those where the distance d between the points Pi and Pf satisfies the condition of non-intersection of the four circles above, $\{Cil \cup Cir\} \cap \{Cfl \cup Cfr\} = \emptyset$. This covers all cases when $d > 4\rho$ and some cases when $d < 4\rho$.

The central idea developed in this study is that the problem of finding the shortest path between two configurations can be reduced to a logical manipulation of the set of appropriate path candidates, without their explicit calculation. This is in sharp departure from the direct computation and comparison of the candidate paths that the existing techniques require. The candidate paths come from the sufficient set known as the Dubins set. One direct benefit of the suggested scheme is computational savings — an important consideration in real-time control. For example, when attempting to find the shortest path to a given position/orientation (configuration) for a driverless car or a mobile robot, one would simply find, the element that corresponds to the initial and final configurations, and then pinpoint the unique solution either immediately or using the sign of an appropriate switching function of the form (Caves, 2010; Hu et al., 2020).

The derivation of the approach is simplified if the problem at hand is divided into two cases, called here th lo e ng path case and the short path case. To save space, the suggested logical classification scheme is fully developed here only for the long path case. Situations with short paths require a roughly similar, though a bit tedious, analysis, resulting in a computational procedure that is somewhat more complex and less economical than the one presented. The presented result also gives an interesting new insight into the nature of Dubins' problem. It suggests that partitioning of the appropriate C e-space can be a powerful tool for analysing the shortest path problem in more general and complex cases, as e.g. in finding the shortest path between a point and a manifold (Karaman & Frazzoli, 2013; Račinský, 2016).

Research Method

Due to differential constraints, the basic form of RRT algorithm for path planning is unusable. The main reason is because Dubin's car is unable to rotate in one place, so it's not possible to reach all configurations from specific state. One of the solutions to modify algorithm to meet

those constraints is using Dubin's curve when building search tree. It means that every branch in this case represents Dubin's curve instead of straight line (Du et al., 2018).

RRT* Algorithm

According to Noreen et al , RRT* is based in a group of features which allow tree expansion very similar to RRT algorithm (Adiyatov & Varol, 2017; Cortés et al., 2007). The difference between the systems is that RRT* incorporates two special properties called near neighbour search and rewiring tree operations. The algorithm is represented by a tree denoted as T = (V, E), where V is a set of vertices and E is a set of edges. The initial configuration (qinit) includes a set of vertices that represents where the tree starts. In each iteration configured (Lines Implementation of Dubin Curves-Based RRT*.

Algoritma RRT *

T = (V, E) RRT *

```
1.  for i = 1, . . . , N do
2.      xrand ← Sample;
3.      xnear ← Near(V, xrand);
4.      (xmin, σ min) ← ChooseParent(Xnear, xrand);
5.      if CollisionFree( σ ) then
6.              V ← V ∪ {xrand};
7.              E ← E ∪ {(xmin, xnew)};
8.              (V, E) ← Rewire( (V, E), Xnear, xrand );
9.  return G = (V, E);
```

The algorithm establishesar and omposition (qrand) infreeregion, (qnearest) is searched in the tree according to predefined step size from (qrand) and immediately the algorithm establishes a new configuration (qnew) with Steer function which guides the system from (qrand) to (qnearest) (Tak et al., 2019). The function Chooseparent allows to select the best parent node for the new configuration (qnew) before its insertion in tree considering the closest nodes that lie inside of a circular area, finding (qmin). Finally, near neighbor operations allow to generate the optimal path repeating the previous process (An et al., 2017; Donsky & Wolfson, 2011).

Dubin Curvess

The Dubin curves describe six types so trajectories: RSR, LSR, RSL, LSL, RLR, and LRL. Each configuration comes from an analogy that is denoted by R (right move), S (straight move), and L (left move)]. All the second figurations use geometrically computing method based on constructing tangent lines between two circles (Lin & Saripalli, 2014). The first step has two circles C1 and C2, with their respective radiusr1 andr2, where C1 represents coordinates (x1, y1) and C2 as (x2, y2) (see in Fig. 1).

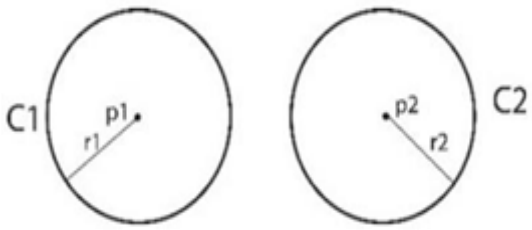

Figure 1 Circles initial configuration

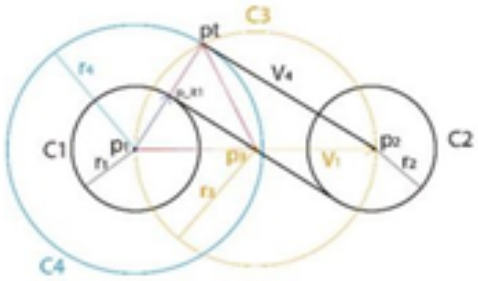

Figure 2 Inner tangents

Inner tangents. Then a line is drawn between two center points C1 and C2 establishing a vector V1, magnitude D and the mid-point p3 (point between p1 and p2) is calculated, circle C3 is constructed with a radius r3 as we reflected in Fig. 2.

$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$
$$p3 = \left(\frac{x1+x2}{2} + \frac{y1+y2}{2}\right) \quad (1)$$
$$r3 = \frac{D}{2}$$

The next step is to draw another circle C4 located in C1's center, with radius r4 = r1 +r2, we obtained pt, which is the intersection between C4 and C3 like the one shown in Fig. 3. A triangle is built joining the points p1, p3, pt and we can define geometrically that segment pt p1 = r4 and p1p3 = pt p3 = r3. The angle γ = ∠pt p1p3 is very important to define the coordinates of pt. The next equation determinates the amount of rotation about the x-axis, θ for V2.

$$\theta = \gamma + \text{atan2}\, V1 \quad (2)$$

Using θ, it is possible to obtain pt according to the following equations:

$$xt = x1 + (r1 + r2) * \cos(\theta)$$
$$yt = y1 + (r1 + r2) * \sin(\theta) \quad (3)$$

Considering that the inner tangent starts on C1, it is necessary to normalize a vector V2 = (pt – p1) and multiply it byr1, the result will allow to find a vector V3 to pit1 from p1. It resumes next by:

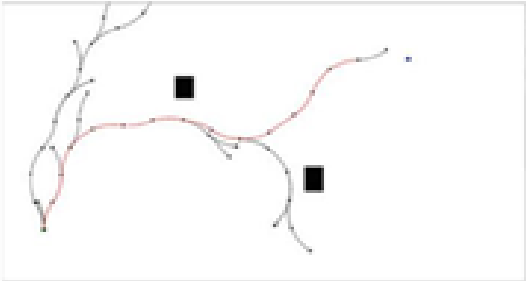

Figure 3 RRT* algorithm's visualization

$$V3 = \frac{V2}{V2} * r1$$
$$pit1 = p1 + V3 \quad (4)$$

Finally, it is possible to draw a vector V4 from pt to p2, as shown in the figure. Using V4 magnitude and the direction, it is possible to find the inner tangent point on C2.

$$V4 = (p2 - pt)$$
$$pit2 = pit1 + V4 \quad (5)$$

Outer tangents. The process is very similar to the one of inner tangents, having two circles C1 and C2, and consideringr1 ≥r2, the procedure is the same as before, C4 is centered at p1, with a difference the radius r4 = r1 –r2, after getting pt and following all steps performed for the interior tangents, V2 is obtained and the first outer tangent point pot1. This condition produces that r4 < r1. To get the second outer tangent pot2 an addition is performed by:

$$pot2 = pot1 + V4 \quad (6)$$

The main difference between calculating outer tangents compared to inner tangents is the construction of circle C4; all steps keep the same. In the next figure (see Fig. 4), it can be seen the path establishes using data input.

A* Algorithm

The A* algorithm was introduced by Hart et al and is currently widely used in the field of shortest path finding problems as a heuristic search algorithm owned by the A* algorithm by evaluating nodes through heuristic search, which improves node search efficiency and has

good performance and accuracy. The A* algorithm evaluates undiscovered places to visit, using a heuristic that estimates the total cost of traveling from origin through a node to a destination by evaluating the most promising node first. A* has advantages such as simple implementation, very short search time, and high efficiency, and can overcome the problem of convergence of basic RRT and lower cost rates.

A* Algorithm

(Start, Goal)

1. Closed-set = the empty set
2. Open-set = includes start node
3. G [Start] = 0, H [Start] H_calc [Start, Goal],
4. F [Start] = H [Start]
5. While Open-set≠ 0 Do
6. CurNode ← EXTRACT-MIN-F (open-set)
7. If (CurNode == Goal), then return Best Path
8. For each Neighbor Node N of CurNode
9. If (N is in Closed-set), Then nothing
10. Else If (N is in Open-set)
11. Calculate N's G, H, F
12. If (G [n ON THE Open set]> calculated G[N])
13. RELAX (N, Neighbor in Open-set, w)
14. N's parent=curNode & add N to open set
15. Else Then calculate N's G, H,F

Dubin Curve RRT*-A*

The Dubin Curve RRT*-A* method is proposed based on the first two fafes which can be represented simply, by forming a tree in the configuration space, starting from the starting point representing the root of the tree, expanding several branches gradually to explore the whole environment and reach the destination point. The main idea of Dubin Curve-RRT*-A* is based on two observations: the value of the radius parameter has a significant impact on the computation time and some nodes of the optimal path are adjacent to the obstacle.

The previous algorithm obtained from the initial solution quality is improved by increasing the radius *Rnear* parameter, because the radius can reduce the cost *Xrand*. When exponential time is the computation time increasing the number of nearest nodes. When the computation time with the radius parameter is left in the Dubin Curve-RRT* when adding to the tree. is added to the tree selected to be abandoned by Dubin Curve-RRT* through two steps namely FindReachest and CreateNode. The Find Reachest procedure searches for the parent where the connected points do not collide. The Create Node procedure to create a node, the Dubin Curve-RRT* algorithm needs to introduce the D parameter to determine the point created

according to the requirements. The following is the Dubin Curve RRT* algorithm process below.

Input: $xstart$, $Xgoal$, Map, $nrepeat$, $Rnear$, $Ddichotomy$

Ouput: G = (V,E)

1. V ← {$xstart$}, E ← ∅;
2. For i = 1 to $nrepeat$ do 3. $xrand$ ← SampleFree(i);
4. $xnearest$ ← Nearest (V, $xrand$);
5. if CollisionFree($xrand$, $xnearest$) then
6. $Xnear$ ← Near (V, $xrand$, $near$);
7. $xreachest$ ← FindReachest (G, $xnearest.xrand$);
8. $xcreate$ ← CreatNode (G, $xreachest$, $xrand$);
9. if $xcreate \neq \emptyset$ then
10. V ← V ∪ {$xcreate$, $xrand$};
11. E← E ∪ {(Parent($xreachest$), $xcreate$), ($xcreate$, $xrand$,)};
12. else
13. V←/ V ∪ {$rand$}.
14. E← E ∪ {($xreachest$, $xrand$)}
15. else
16. if InitialPathFound then
17. Return G= (V, E);
18. end
19. G← Rewire (G, $xrand$, $Xnear$);
20. end
21. end
22. Return G= (V, E);

System Block Diagram

In this sub-chapter, block diagrams of systems are described, with each block interconnected. Block diagrams have several functions, namely: explaining how a system works in a simple way, analyzing how the system works, and making it easier to check for errors in the system being built. The block diagram of the system to be designed is as shown in Figure 3.1

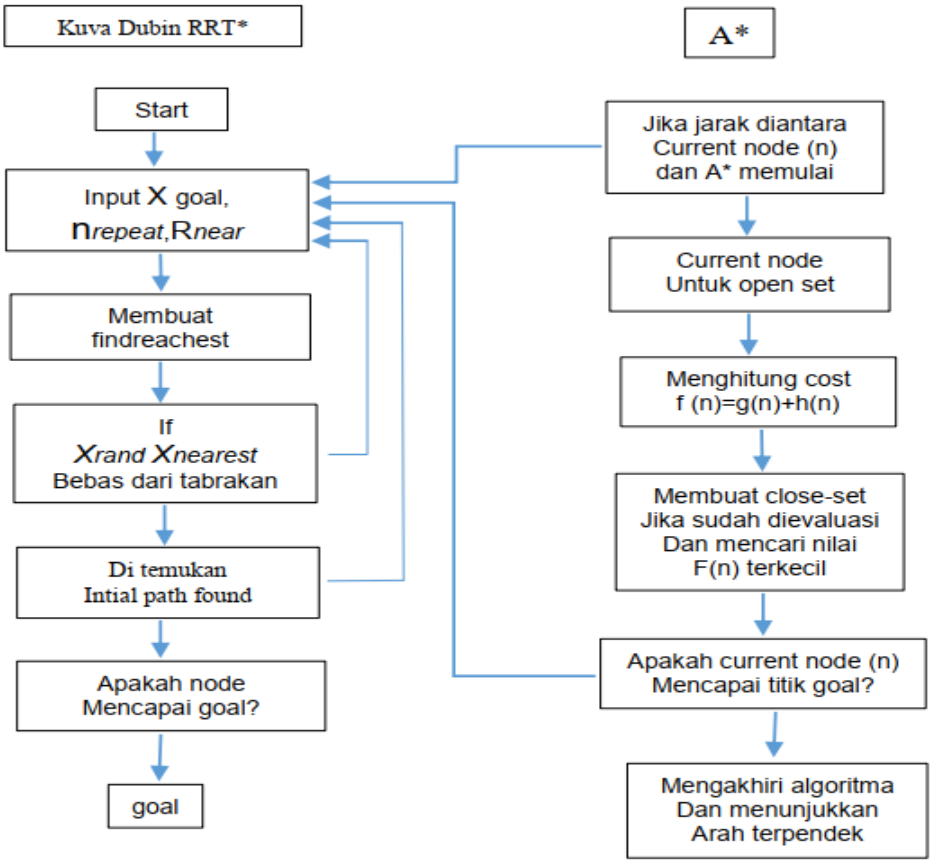

Figure 4 System Block Diagram

Result and Discussion

The Result

The results of the design of the RRT*, A*, and Dubin Curve-RRT*-A* algorithms are three maps measuring 70 x 70 cm. The results of this test explain the results of the test, the path planning simulation is divided into three: Group-1 consists of three experimental groups RRT*, A*, and Dubin Curve-PRC*-A* experiments are applied using a predetermined map. In matlab the test results are seen in terms of the optimization path and the time taken is the shortest time with the three maps that have been made. Previously the method was tested by implementing it in various scenarios, then the results were compared, and the performance evaluated depending on several factors (cost, convergence). With parameter on map 1, Start point x,y= (8,25); Goal points x,y=(40,50); Maximum nodes: 5000; Step size=1;Goal threshold=<10 (distance between A* points);Resolution = 1. In the second map, the difference is only at the start point and end point. Start point x,y= (8,25); Goal points x,y=(25,40). In the third map, it differs only in the start point and end point. Start point x,y= (8,25); Goal points

x,y=(35,45). The number of obstacles is different between the two maps. Each of these maps is used with the previous method to plot the path of the robot. The results of the search operations of all experiments are discussed next at the end of the section.

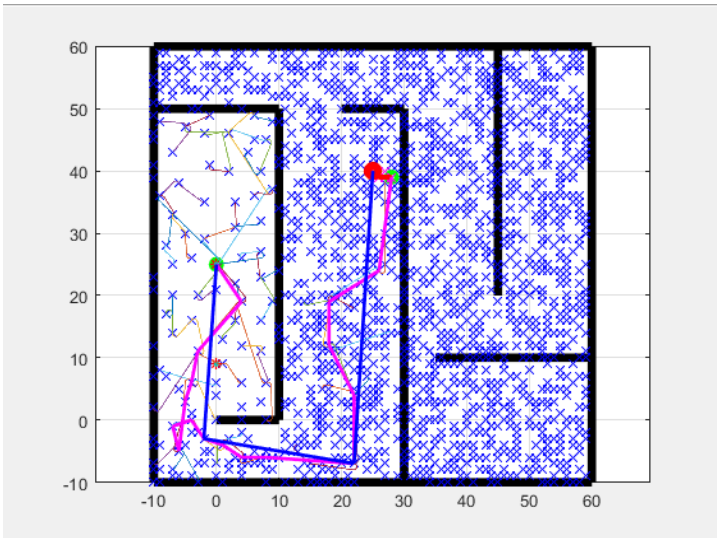

Figure 5 Test Results of the PRC Dubin Curve RRT*-A

The research of each comparison simulation of the three methods resulted in the optimization of Dubin's improvement path and the time used. In the results of the RRT* algorithm image that the sampling points are evenly distributed throughout the room, then RRT* uses more nodes to cover the entire room, for A* because it is searching based this algorithm evaluates the entire room to reach the goal point so it takes a long time , the three Dubin Curve-RRT*-A* algorithms sampling points of the Enhanced RRT algorithm are more densely distributed and the random tree is more efficient to reach the state space. The mean and variance were used as two indicators for the characterization of the search time. The mean seek time value represents the average performance of the algorithm, while the variance indicates its stability, i.e. both are very important for practical applications (e.g., robots).

Comparison Result

Based on the results of the visual research above, the authors propose and evaluate to find a feasible path in the map. The algorithm needs to find a gap through the obstacles from the start point to the end point. Due to the requirements for sampling in large quantities. Several experimental groups were carried out by taking five samples on map 1 of each RRT*, A*, and Dubin Curve-RRT*-A* algorithms, then the mean and variance of the algorithm search time were calculated.

Table 1 Comparison Result Testing

Method	Time				
	Step 1	Step 2	Step 3	Step 4	Step 5
RRT*	54,729	27,51	150,73	37,18	62,52
A*	120,47	150,84	100,38	115,67	98.79
Dubin Curve RRT*-A*	33.68	25.14	20,27	27.68	24.56

In the map 1 experiment, we can see in table 4.1 that Dubin Curve RRT*-A* has the lowest time compared to RRT* and A*. That means it can be concluded that Dubin Curve RRT*-A* is more suitable for use for more optimal robot paths.

Conclusions

The conclusion of the research Merging the Dubin Curve-RRT* and A* algorithms to find the optimal path by producing a node curve near the obstacle is the proposed path planning algorithm, namely Dubin Curve- RRT* which is to increase the speed and stability in finding the initial path. Adding a new algorithm to find a new path optimally to get a path that is close to optimal by combining and adjusting several feasible paths and also adding a searching-based algorithm, namely A* combined with Dubin Curve- RRT* which is sampling based, where the A* algorithm has a heuristic function which is used to increase the cost and time of the search. By checking each Dubin Curve-RRT* node to check if it belongs to the destination region with a certain threshold. When it reaches this region or radius then the A-star algorithm is activated. The goal is to optimize the shortest path to the goal point and make it smoother by using some sort of regression. The result proves that the combination process used in the proposed method has contributed to the reduction of the cost of PRC* and the time level (convergence).

References

Adiyatov, O., & Varol, H. A. (2017). A novel RRT*-based algorithm for motion planning in dynamic environments. 2017 IEEE International Conference on Mechatronics and Automation (ICMA),

An, B., Kim, J., & Park, F. C. (2017). An adaptive stepsize RRT planning algorithm for open-chain robots. *IEEE Robotics and Automation Letters*, *3*(1), 312-319.

Caves, A. D. J. C. C. (2010). *Human-automation collaborative RRT for UAV mission path planning* Massachusetts Institute of Technology].

Cortés, J., Jaillet, L., & Siméon, T. (2007). Molecular disassembly with RRT-like algorithms. Proceedings 2007 IEEE International Conference on Robotics and Automation,

Daniel Tenezaca, B., Canchignia, C., Aguilar, W., & Mendoza, D. (2020). Implementation of dubin curves-based RRT* using an aerial image for the determination of obstacles and path planning to avoid them during displacement of the mobile robot. Developments and Advances in Defense and Security: Proceedings of MICRADS 2019,

Donsky, E., & Wolfson, H. J. (2011). PepCrawler: a fast RRT-based algorithm for high-resolution refinement and binding affinity estimation of peptide inhibitors. *Bioinformatics*, *27*(20), 2836-2842.

Du, Z., Wen, Y., Xiao, C., Zhang, F., Huang, L., & Zhou, C. (2018). Motion planning for unmanned surface vehicle based on trajectory unit. *Ocean Engineering*, *151*, 46-56.

Hu, B., Cao, Z., & Zhou, M. (2020). An efficient RRT-based framework for planning short and smooth wheeled robot motion under kinodynamic constraints. *IEEE Transactions on Industrial Electronics*, *68*(4), 3292-3302.

Jacobs, P., Laumond, J. P., & Taix, M. (1991, 3-5 Nov. 1991). Efficient motion planners for nonholonomic mobile robots. Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91,

Karaman, S., & Frazzoli, E. (2013). Sampling-based optimal motion planning for non-holonomic dynamical systems. 2013 IEEE international conference on robotics and automation,

Lin, Y., & Saripalli, S. (2014). Path planning using 3D dubins curve for unmanned aerial vehicles. 2014 international conference on unmanned aircraft systems (ICUAS),

Račinský, M. (2016). *Aplikace algoritmu RRT-path v úloze autonomního dohledu skupinou helikoptér* České vysoké učení technické v Praze. Vypočetní a informační centrum.].

Soueres, P., & Laumond, J. P. (1996). Shortest paths synthesis for a car-like robot. *IEEE Transactions on Automatic Control*, *41*(5), 672-688. https://doi.org/10.1109/9.489204

Tak, H.-T., Park, C.-G., & Lee, S.-C. (2019). Improvement of RRT*-smart algorithm for optimal path planning and application of the algorithm in 2 & 3-dimension environment. *Journal of the Korean Society for Aviation and Aeronautics*, *27*(2), 1-8.

Tenezaca, B. D., Canchignia, C., Aguilar, W., & Mendoza, D. (2019). Implementation of Dubin Curves-Based RRT* Using an Aerial Image for the Determination of Obstacles. *Developments and Advances in Defense and Security: Proceedings of MICRADS 2019*, *152*, 205.