Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 3, No. 1, March : 2023

OBSTACLE AVOIDANCE FOR UAV PATH USING OPTIMIZED GRADIENT DESCENT ALGORITHM

Mrs. N. MALATHI[1], K. NAGA INDU SRI[2], K. PRAVALIKA[3], K. TANUJA[4], K. TANVI[5]
[1] M. Tech, Assistant Professor of ECE, NRI Institute of Technology
[2345] B. Tech, Student, Department of ECE, NRI Institute of Technology

ABSTRACT

UAVs are becoming more useful and common in agriculture as well as the daily life applications. Path planning is intended to control the motion of the Unmanned Aerial Vehicles in a specific trajectory smoothly and quickly. In UAV-to-UAV communication obstacle consideration plays an important role. In obstacle avoidance path planning capabilities of UAV that would go over safe zones most of the time. To avoid obstacles existing approaches for path planning are necessary to frequently detect whether two objects estimate the shortest between source and destination. Moreover, collision checking and minimum distance estimating themselves are complex and time-consuming tasks. To deal with the above-mentioned problems for path planning with obstacle avoidance we use Gradient Descent Algorithm.

Keywords: *Unmanned aerial vehicles, Trajectory, Obstacles, Gradient Descent Algorithm.*

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are capable of flying autonomously or through different control modalities such as joysticks, smartphones, voice and gestures. Until the early 2000s, UAVs were mainly used in the military area, with the development in hardware and software technologies UAVs got smaller, easier to control, and less costly. Now, UAVs can perform a wide range of civilian activities. Among these activities, UAVs are firstly used for entertainment, such as for photography during extreme sports activities. Then it becomes possible for UAVs to help people to finish work in many different ways. In some situations, tasks might be hard or harmful for humans. UAVs are useful in these situations and can improve efficiency. With an overhead view of working sites, UAVs can be used for surveillance and reporting progress. In such scenarios, moving objects can exist together with UAVs. This brings challenges to the operation of UAVs.

To operate UAVs automatically in dynamic environments, it is crucial for UAVs to avoid static obstacles and dynamic obstacles in a safe way while executing tasks. When UAVs are working, the requirements are to be aware of working environments, to be able to predict obstacle movement in particular situations, and to be able to avoid colliding with static and moving obstacles. The challenge is to achieve real-time navigation in dynamic environments which change unpredictably. This challenge relates to the area of motion planning. Therefore, to solve this challenge, a hybrid method which integrates benefits of both motion planner and trajectory optimizer is required. Furthermore, to develop and test motion planning algorithms in a safe and inexpensive manner, UAV simulators and UAV controllers are required for running simulations. The challenges in this area are designing realistic scenarios, simulating UAV behaviours and visualizing generated trajectories.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 3, No. 1, March : 2023

II. RELATED WORK

This section gives a review of relevant literature. The optimized motion planning framework consists of optimized motion planner and dynamic scene generator. When designing an optimized motion planner, the related areas are motion planning algorithms and trajectory optimization. So, the first part is an introduction to motion planning algorithms in static and dynamic environments, and the second part describes related work in trajectory optimization. Dynamic scene generator module utilized UAV simulator. The third part describes related work in UAV simulation.

Motion Planning

In the UAV motion planning area, large numbers of state-of-the-art algorithms have been proposed in the last few decades. Research in this area was initially focused on planning in static environments, and from there expanded towards dynamic environments, which take into consideration uncontrollable agents as well as moving obstacles. More algorithms are proposed to optimize the planned trajectories, reduce the computation cost and be more robust to model uncertainty or disturbances.

Static Environments

In static environments, obstacles are stationary. With a map known a priori, the configuration space consists of free space. When the size and complexity of the map increase, the number of grids will increase, so the time cost and the memory usage will increase severely. Then the random sampling methods are proposed. In 1996, probabilistic roadmaps (PRM) were proposed. PRM is a graph of possible paths in a space which has free and occupied parts. It can be used to find the path with a relatively small number of samples nodes, but cannot make sure to find the optimal path. When exploring a larger space or in high dimensional space, the time and computation cost will be high. But when obstacles are moving, the planned path is required to be updated as well. To solve this problem, many algorithms are developed for motion planning in dynamic environments.

Dynamic environments

Daily common environments are complicated and dynamic. To navigate in the dynamic environment, UAV need to consider the movement of the obstacles, then plan a feasible and optimal trajectory. To achieve real-time UAV navigation in a large space, motion planning algorithms need to be improved with high efficiency and relatively low storage requirement. In 2006, Ferguson proposed Dynamic RRT (DRRT) algorithm will remove invalid nodes, reconnect and regrow to repair the solution path. During this procedure, the obstacles are still considered stationary. To increase the efficiency and make use of more information of previous planned path in the replanning procedure. Zhu et al proposed an algorithm which uses the depth camera to get the positions and velocities of the moving obstacles to generate a cost map and integrate it into the planning procedure. The cost map will be used to determine potential collision. The limitation of this method is that it assumes all obstacles are moving at a constant speed.

Trajectory Optimization

Besides collision avoidance, another objective of the motion planning framework is achieving smooth and efficient trajectories. To improve the quality of the trajectory, trajectory optimization techniques can be utilized. Trajectory optimization algorithms can be used to smooth and shorten trajectories generated by other methods. And they can be used to plan from scratch, which initializes with a trajectory that contains collision and perhaps violates constraints and then converges to a high-quality trajectory satisfying constraints. Obstacles are considered directly in the workspace of the UAV, where the notions of distance and inner product are more natural.

UAV Simulator

In the study of UAV motion planning, simulator is important. It allows evaluating algorithms in a safe and inexpensive manner, without worrying about dealing with real-world hardware. The ideal simulator needs to be fast, physically accurate and photo-realistic. UAV simulator consists of UAV control system and UAV simulation. UAV simulator module contains four nodes, which are: start quadrotor, autopilot, unity scene and path visualization. UAV control system and UAV simulation are both constructed based on existing software.

III. PROPOSED METHODOLOGY

While a lot of attention was given to motion planning algorithms development and simulation separately, there is little work focusing on integrating motion planner and realistic simulator. The challenge is to find the shortest path i.e., path planning. In this thesis, Gradient Descent Algorithm is designed and developed. The work is motivated based on the need for searching a hybrid method which integrates motion planner and trajectory optimizer. The framework enables a UAV to navigate autonomously and safely in dynamic environments. Safety is demonstrated by ensuring that the UAV does not collide with any obstacles during the entire navigation. The framework consists of three parameters. After finding a trajectory from the current position of the UAV to the goal, an optimizer is applied to optimize the trajectory. This module aims to generate a collision-free trajectory and be able to modify it to avoid collisions when environment changes. Dynamic scene generator, which contains obstacle information messenger and UAV simulator. This module is to generate Unity scene with static and dynamic obstacles, then UAV, obstacles and trajectories generated by optimized motion planner will be visualized in the same scene. UAV simulator utilizes Frightmare, which is a flexible modular quadrotor simulator that contains rendering engine built on Unity and physics engine for dynamics simulation. Within UAV simulator, UAV control system utilizes RPG quadrotor control library which contains a position controller and a state machine. The framework is an open-source project which is released under the MIT license.

In the proposed system we use Improved Gradient Descent Algorithm , to make the path planning asymptotically optimal. This algorithm can set pheromones on the path obtained by Gradient Descent Algorithm and select the next extension point according to the pheromone concentration. Through a certain number of iterations, it converges into an ideal path scheme. Specifically, this algorithm is formulated to minimize the energy, time and distance. Simulation results show that the Gradient Descent Algorithm is more efficient compared with a data transmission based gradient algorithm. Therefore by using this proposed algorithm we can minimize the path length from one position to another position that is from source to destination.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 3, No. 1, March : 2023

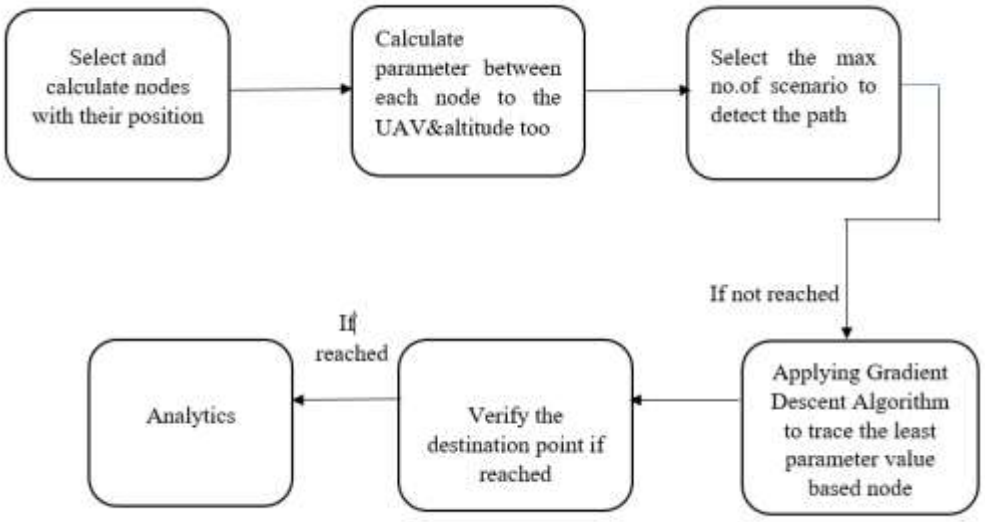

Fig 1. Block Diagram

Proposed Algorithm – Gradient Descent Algorithm

Gradient descent (GD) is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning (DL) to minimise a cost/loss function (e.g. in a linear regression). Due to its importance and ease of implementation, this algorithm is usually taught at the beginning of almost all machine learning courses.
However, its use is not limited to ML/DL only, it's being widely used also in areas like:

- control engineering (robotics, chemical, etc.)
- computer games
- mechanical engineering

That's why today we will get a deep dive into the math, implementation and behaviour of first-order gradient descent algorithm. We will navigate the custom (cost) function directly to find its minimum, so there will be no underlying data like in typical ML tutorials — we will be more flexible in terms of a function's shape.

This method was proposed before the era of modern computers and there was an intensive development meantime which led to numerous improved versions of it but in this article, we're going to use a basic/vanilla gradient descent implemented in Python

Function requirements

Gradient descent algorithm does not work for all functions. There are two specific requirements. A function has to be:

- **differentiable**
- **convex**

3.3 Gradient

Before jumping into code one more thing has to be explained — what is a gradient. Intuitively it is a slope of a curve at a given point in a specified direction.

In the case of **a univariate function**, it is simply the **first derivative at a selected point**. In the case of **a multivariate function**, it is a **vector of derivatives** in each main direction (along variable axes). Because we are

interested only in a slope along one axis and we don't care about others these derivatives are called **partial derivatives**.

A gradient for an n-dimensional function f(x) at a given point p is defined as follows:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

The upside-down triangle is a so-called *nabla* symbol and you read it "del". To better understand how to calculate it let's do a hand calculation for an exemplary 2-dimensional function below.

$$f(x) = 0.5x^2 + y^2$$

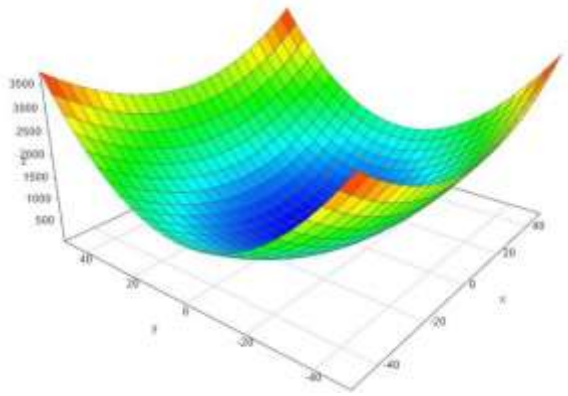

Figure 3.3: 3D plot of Gradient

Let's assume we are interested in a gradient at point p(10,10):

$$\frac{\partial f(x,y)}{\partial x} = x, \quad \frac{\partial f(x,y)}{\partial y} = 2y$$

so consequently:

$$\nabla f(x,y) = \begin{bmatrix} x \\ 2y \end{bmatrix}$$

$$\nabla f(10,10) = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

By looking at these values we conclude that the slope is twice steeper along the y axis.

3.4 Gradient Descent Algorithm

Gradient Descent Algorithm iteratively calculates the next point using gradient at the current position, scales it (by a learning rate) and subtracts obtained value from the current position (makes a step). It subtracts the value because we want to minimise the function (to maximise it would be adding). This process can be written as:

$$p_{n+1} = p_n - \eta \nabla f(p_n)$$

There's an important parameter **η** which scales the gradient and thus controls the step size. In machine learning, it is called **learning rate** and have a strong influence on performance.

- The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point
- If learning rate is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

In summary, Gradient Descent method's steps are:

1. choose a starting point (initialisation)
2. calculate gradient at this point
3. make a scaled step in the opposite direction to the gradient (objective: minimise)
4. repeat points 2 and 3 until one of the criteria is met:
5. maximum number of iterations reached
6. step size is smaller than the tolerance (due to scaling or a small gradient).

Below, there's an exemplary implementation of the Gradient Descent algorithm (with steps tracking):
This function takes 5 parameters:
1. **starting poin**t - in our case, we define it manually but in practice, it is often a random initialisation
2. **gradient functio**n - has to be specified before-hand
3. **learning rate** - scaling factor for step sizes
4. maximum number of iterations
5. tolerance to conditionally stop the algorithm (in this case a default value is 0.01)

Example 1 — a quadratic function

Let's take a simple quadratic function defined as:

$$f(x) = x^2 - 4x + 1$$

Because it is an univariate function a gradient function is:

$$\frac{df(x)}{dx} = 2x - 4$$

Let's write these functions in Python:
For this function, by taking a learning rate of 0.1 and starting point at x=9 we can easily calculate each step by hand. Let's do it for the first 3 steps:

$$x_1 = 9 - 0.1 \cdot (2 \cdot 9 - 4) = 7.6$$
$$x_2 = 7.6 - 0.1 \cdot (2 \cdot 7.6 - 4) = 6.48$$
$$x_3 = 6.48 - 0.1 \cdot (2 \cdot 6.48 - 4) = 5.584$$

The python function is called by:

The animation below shows steps taken by the GD algorithm for learning rates of 0.1 and 0.8. As you see, for the smaller learning rate, as the algorithm approaches the minimum the steps are getting gradually smaller. For a bigger learning rate, it is jumping from one side to another before converging.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 3, No. 1, March : 2023

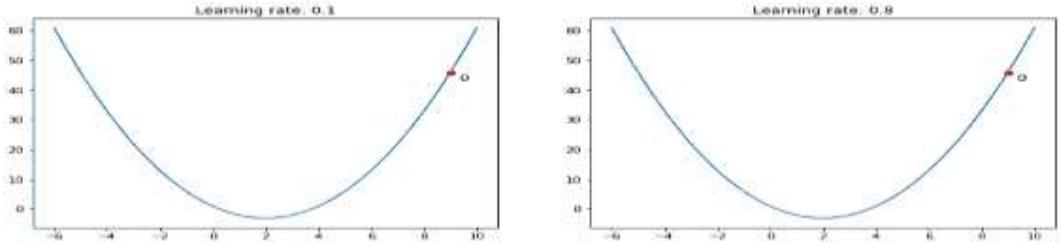

Figure 3.4: Results for learning rate

First 10 steps taken by GD for small and big learning rate

Trajectories, number of iterations and the final converged result (within tolerance) for various learning rates are shown below:

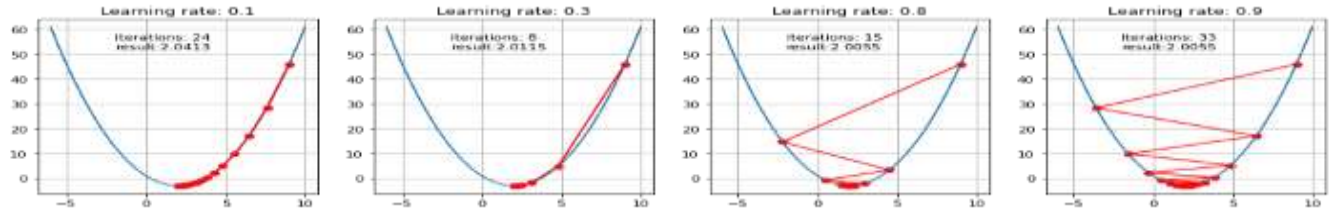

Figure 3.4.1: Results for various learning rates

Example 2 — a function with a saddle point

Now let's see how the algorithm will cope with a semi-convex function we investigated mathematically before.

$$f(x) = x^4 - 2x^3 + 2$$

Below results for two learning rates and two different staring points.

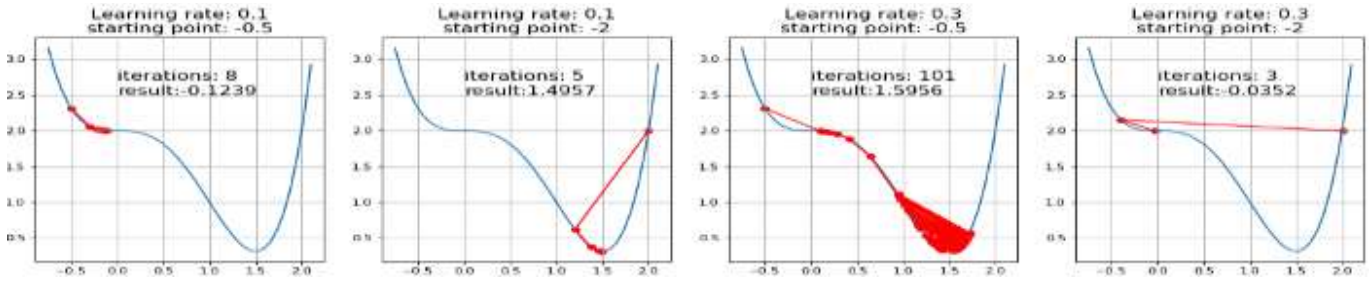

Figure 3.4.2: GD trying to escape from a saddle point

Below an animation for a learning rate of 0.4 and a starting point *x=-0.5*..

IV. **RESULT**S ANALYSIS

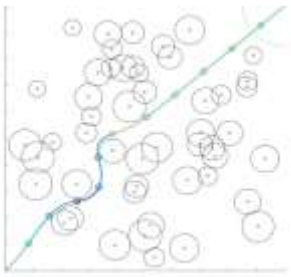

Figure 10.3:Result 1

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 3, No. 1, March : 2023

This is the Result 1 it indicates the final shortest final path with obstacle avoidance from source to destination.Here the circle indicates the obstacles and the right top curve represents the destination i.e.,the end point of UAV. Here in result 1 it consists of path segments.Here the UAV find the shortest path for better results and for fast transfer of information.

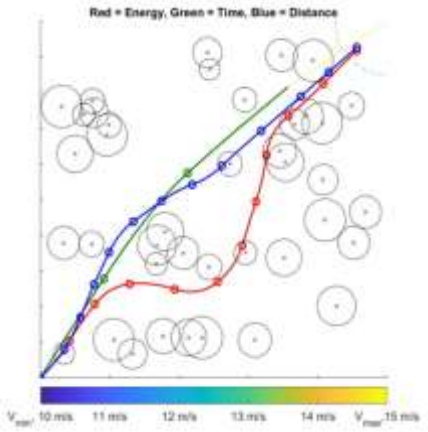

Fig Result 2

Result 2 represents the framework that minimizes the Time, Distance paths from initial source node UAV to destination node UAV. The velocity is varying from Vmin=10m/s to Vmax=15m/s in between these the dynamic obstacles are generated randomly. The red line shows energy line is required because by searching multiple numbers of paths and choosing shortest path, the green line shows time line having less amount of time is required to transmit the data from initial point to final point. The blue line shows distance line having more number of planned path segments are considered to avoid a dangerous obstacles. The multiple numbers of planned path segment points can observe on blue line.

V. CONCLUSION & FUTURE ENHANCEMENT

The main goal of this work is to create the short path for UAV to UAV Trajectory that would go over safe zones most of the time. Path planning is intended to control the motion of the Unmanned Aerial Vehicles in a specific trajectory smoothly and quickly. Considered the positions of the UAVs are varied and the Trajectory also varied, it is not constant and velocity is also varying from 10msec to 15msec. The displacement of UAVs will change the transmission rate and energy consumption. In this article, Gradient Descent Algorithm was proposed.Therefore, with the help of proposed path planning optimization algorithm we minimized the path length from one position to another destination while avoiding concussion with different obstacles. Specifically, the proposed optimization algorithm is formulated to minimize the energy, time and distance and also the delay between two nodes. Furthermore, there are also many changes to implement the path planner into a functional method.

By utilizing this proposed framework, more advanced Gradient Descent algorithms can be evaluated and simulated. The best path planning can be improved in more realistic and complex places. The prediction of obstacles can be improved to help UAV to navigate in a more complicated environment and avoid obstacles more precisely. In the evaluation step, limited scenarios are included. So the interface of obstacle info messenger can be improved to simulate a wider variety of obstacles so in future enhancement better version of Gradient Descent

Algorithm will be developed for better path planning and obstacle avoidance with more parameters. The Gradient Descent algorithm can be improved to consider avoidance of humans and provide a solution for safe human-drone interaction in dynamic environments. In this future enhancement more parameters can derive in this path planning.

REFERENCES

[1]P. G. Fahlstrom and T. J. Gleason, Introduction to UAV Systems, Fourth Edition. John Wiley & Sons, Ltd, 2012. ISBN 9781118396780 [Page 1.]

[2]P. Springer, Military Robots and Drones: A Reference Handbook, ser. Contemporary world issues. ABC-CLIO, 2013. ISBN 9781598847321 [Page 1.]

[3]M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," The International Journal of Robotics Research, vol. 35, no. 7, pp. 797–822, 2016. doi: 10.1177/0278364915594679 [Pages 2, 7, 13, and 19.]

[4]M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith,C.Dellin, J. A. D. Bagnell, and S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," International Journal of Robotics Research, vol. 32, no. 9, pp. 1164 – 1193, August 2013. [Pages 2, 9, 14, and 15.]

[5]S. Choudhury, J. D.Gammell, T. D. Barfoot, S. S. Srinivasa, and S.Scherer,"Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016. doi: 10.1109/ICRA.2016.7487615 pp. 4207–4214. [Page 9.]

[6]P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968. doi: 10.1109/TSSC.1968.300136 [Page 11.]

[7]E. W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math., vol. 1, no. 1, p. 269–271, Dec. 1959. doi: 10.1007/BF01386390 [Page 11.]

[8]L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Prob-abilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996. doi: 10.1109/70.508439 [Page 11.]

[9]S. LaValle, "Rapidly-exploring random trees : A new tool for path planning," The annual research report, 1998. [Page 11.]

[10]J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in 2000 IEEE International Conference on Robotics and Automation (ICRA), vol. 2, 2000, pp. 995–1001 vol.2. [Page 11.]

[11]S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, vol. 30, no. 7, pp. 846–894, 2011. doi: 10.1177/0278364911406761 [Page 11.]

[12]O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in 2013 IEEE International Conference on Mechatronics and Automation (ICMA), 2013. doi: 10.1109/ICMA.2013.6617944 pp. 354–359. [Page 12.]

[13]D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in 2006 IEEE International Conference on Robotics and Automation (ICRA), 2006. doi: 10.1109/ROBOT.2006.1641879 pp. 1243–1248. [Page 12.]

[14]A. Stentz, “Optimal and efficient path planning for partially-known environments,” in 1994 IEEE International Conference on Robotics and Automation (ICRA), 1994. doi: 10.1109/ROBOT.1994.351061 pp. 3310–3317 vol.4. [Page 12.]
[15]O. Adiyatov and H. A. Varol, “A novel RRT*-based algorithm for motion planning in dynamic environments,” in 2017 IEEE International Conference on Mechatronics and Automation (ICMA), 2017. doi: 10.1109/ICMA.2017.8016024 pp. 1416–1421. [Page 12.]
[16]L. Zhu, Z. Chi, F. Zhou, and C. Zhuang, “Dynamic motion planning algorithm in human-robot collision avoidance,” in 2019 12th International Conference Intelligent Robotics and Applications (ICIRA), 2019. ISBN 978-3-030-27529-7 pp. 655–666. [Page 12.]
[17]C. Fulgenzi, A. Spalanzani, C. Laugier, and C. Tay, “Risk based motion planning and navigation in uncertain dynamic environment,” Research Report, Oct. 2010. [Page 13.]
[18]W. Chi and M. Q. . Meng, “Risk-RRT*: A robot motion planning algorithm for the human robot coexisting environment,” in 2017 18th International Conference on Advanced Robotics (ICAR), 2017, pp. 583– 588. [Page 13.]
[19]D. Connell and H. M. La, “Dynamic path planning and replan-ning for mobile robots using RRT*,” in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017. doi: 10.1109/SMC.2017.8122814 pp. 1429–1434. [Page 13.]
[20]B. Chandler and M. A. Goodrich, “Online RRT* and online FMT*: Rapid replanning with dynamic cost,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017. doi: 10.1109/IROS.2017.8206535 pp. 6313–6318. [Page 13.]
[21]J. J. Park, C. Johnson, and B. Kuipers, “Robot navigation with model predictive equilibrium point control,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012. doi: 10.1109/IROS.2012.6386195 pp. 4945–4952. [Page 14.]
[22]S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, “FaSTrack: A modular framework for fast and guaranteed safe motion planning,” in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), 2017. doi: 10.1109/CDC.2017.8263867 pp. 1517–1522. [Page 14.]
[23]S. Berchtold and B. Glavina, “A scalable optimizer for automatically generated manipulator motions,” in Proceedings of IEEE/RSJ Interna-tional Conference on Intelligent Robots and Systems (IROS’94), vol. 3, 1994. doi: 10.1109/IROS.1994.407615 pp. 1796–1802 vol.3. [Page 14.]
[24]D. Hsu, J.-C. Latcombe, and S. Sorkin, “Placing a robot ma-nipulator amid obstacles for optimized execution,” in Proceed-ings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP’99) (Cat. No.99TH8470), 1999. doi: 10.1109/ISATP.1999.782972 pp. 280–285. [Page 14.]
[25]R. Geraerts and M. H. Overmars, “Creating high-quality paths for motion planning,” The International Journal of Robotics Research, vol. 26, no. 8, pp. 845–863, 2007. doi: 10.1177/0278364907079280 [Page 14.]