INF421 PI: PATH PLANNING ALGORITHMS

MARCEAU COUPECHOUX

1. Introduction

The purpose of this project is to design, implement, and evaluate path planning algorithms, e.g., for a robot or a UAV. Path planning (or motion planning) consists in "finding a collision-free motion between an initial (start) and a final configuration (goal) within a specified environment" [GBLV15]. In this project, we consider a simple instance of this problem: The environment is a rectangle, made of a free space and obstacles; a robot starts from the lower left corner of the square and should reach the upper right corner while avoiding obstacles along a path of minimum distance. To solve this computational problem, we consider two heuristics, namely Particle Swarm Optimization (PSO), a bio-inspired algorithm, and Rapidly-exploring Random Trees (RRT), a graph-based approach. Many others exist, but these two are simple and easily accommodate models related to robot or UAV constraints. Figure 1 shows an example of path found with RRT.

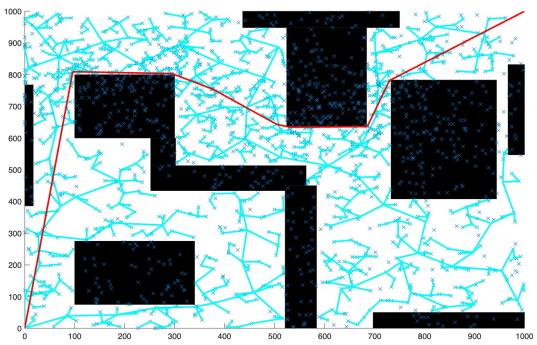

Figure 1. Example of path avoiding obstacles between $(0, 0)$ and $(1000, 1000)$ (in red) and RRT traces (in blue).

2. Setting up the environment

The environment is a rectangle whose lower left corner has coordinates $(0, 0)$ and the upper right corner has coordinates (x_{max}, y_{max}). The starting position of the robot is denoted $u_s^{(1)}$ and the goal is $u_d^{(1)}$. At last, there obstacles which are rectangles characterized by 4-uplets $(x_o, y_o, \ell_x, \ell_y)$, where (x_o, y_o) are the coordinates of the lower left corner of the rectangle, ℓ_x is its dimension along the x-axis and ℓ_y its dimension along the y-axis.

Test files corresponding to 5 scenarios can be downloaded by following this link. The format is the following: x_{max}, y_{max}, $u_s^{(1)}$ (2 coordinates), $u_d^{(1)}$, $u_s^{(2)}$, $u_d^{(2)}$, R and for every obstacle $(x_o, y_o, \ell_x, \ell_y)$. The parameters $u_s^{(2)}$, $u_d^{(2)}$ and R are used only in Section 5.

Question 1. Write a method that takes such a file as input, checks that it is a valid input and initializes a path planning problem. Explain your implementation choices.

Date: Nov. 2025.

Question 2. Write a method that displays the environment and any given path between the starting point and the destination.

3. A first approach: Particle Swarm Optimization

In this section, we consider a bio-inspired algorithm called Particle Swarm Optimization [KE95]. The principle of PSO is to iteratively try to improve candidate solutions, called *particles*. Every particle is able to move within the search-space and is characterized by its position and velocity. Its movement is influenced by its local best known position and guided towards the best known position ever found [wik]. Although there is no guarantee to converge to an optimal solution, PSO has been shown to be an efficient algorithm for path planning problems, see e.g. [AKYT23].

3.1. Basic algorithm.

To go in more details, the algorithm proceeds iteratively and we designate by k the iteration step. Let S be the number of particles. Their positions can be compared thanks to a *fitness* function (also called *objective* function in optimization) that should be minimized. Particle i is characterized by a position x_i^k and a velocity v_i^k at step k. Every particle keeps track of its own best ever seen position (or local best solution, denoted p_i^k for particle i at step k) and the best position ever seen by all particles (or global best solution, denoted g^k at step k). Positions and velocities are updated according to the following formulas:

$$v_i^{k+1} = wv_i^k + c_1 r_1 (p_i^k - x_i^k) + c_2 r_2 (g^k - x_i^k) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2)$$

where c_1 and c_2 are acceleration coefficients, which control the influence of the local and global best solutions. The variables r_1 and r_2 are random variables uniformly drawn in $[0, 1]$. The parameter w is an inertia weight for velocity. Positions and velocity can be further constrained to closed intervals. We call S, c_1, c_2, and w the *hyper-parameters* of the algorithm. After a given number of iterations, or if a stopping condition is met, the algorithm returns the global best position.

Question 3. Define particles and a fitness function for the specific problem of path planning and propose a data structure for particles.

Question 4. Propose a method that checks if a path is colliding with an obstacle.

Question 5. Provide the pseudo-code of your PSO algorithm for path planning.

Question 6. Give the complexity of your algorithm.

Question 7. Implement your algorithm and explain your implementation choices.

Question 8. For every scenario given in the test files, provide the path length output by your algorithm, the number of iterations to obtain this path, and the CPU time. Display the path in the environment. Explain how you have chosen the hyper-parameters and the involved tradeoffs.

3.2. Algorithm improvements.

It is possible to improve the efficiency or the convergence speed of the algorithm with dedicated procedures. The first one is *random restart*: particles are periodically reinitialized to a random value. The goal is to explore zones of the search-space that could have been missed due to the particular choice of the initial positions.

Question 9. Modify your pseudo-code in order to take into account the random restart procedure and compare the results with those obtained with the basic algorithm (in terms of path length and speed of convergence) on the test files. If you introduce new hyper-parameters, explain how their value has been chosen and the involved tradeoffs.

Another improvement is called *simulated annealing*. It tries to get out of local optima for g^k, where the algorithm could be stuck. The procedure depends on a parameter called the *temperature*, denoted T^k at iteration k. Whenever a new candidate solution s is considered, we compute its fitness $f(s)$ and adopt it as a global best solution with probability $\min(1, \exp(-(f(s) - f(g^k))/T)$. The temperature evolves with time as $T^k = T^0 \beta^k$, where T_0 is the initial temperature and $0 < \beta < 1$ controls the rate of decreasing.

Question 10. Modify your pseudo-code in order to take into account the simulated annealing procedure and compare the results with those obtained with the basic algorithm. Simulated annealing can be combined with random restart. Explain how T_0 and β have been set in your experiments and the involved tradeoffs.

Another improvement is called *dimensional learning* [XCS+19]. This procedure tries to reduce a phenomenon of oscillation between local and global best solutions and to accelerate convergence. If a particle has not updated its local best solution during a certain number of steps, its local best solution p_i^k is updated by using the information of the global best solution g^k as follows: the j-th dimension of g^k replaces the j-th dimension of p_i^k if the resulting fitness is lower, j runs along all the dimensions of a position. See [XCS+19] for examples.

Question 11. Modify your pseudo-code in order to take into account the dimensional learning procedure and compare the results with those obtained with the basic algorithm. Dimensional learning can be combined with random restart and simulated annealing. If you introduce new hyper-parameters, explain how their value has been chosen.

Question 12. Propose your own improvement of the algorithm and perform performance comparison with the four proposed scenarios. What do you conclude on the efficiency of PSO to solve the path planning problem?

4. A second approach: Rapidly-exploring Random Tree

We now consider RRT. The algorithm consists in constructing a space-filling tree until the destination is reached.

4.1. Basic algorithm.

At every iteration, the algorithm samples a random point v_r from a set E. Within the already constructed tree, it searches for a closest node v_n. A new node v is created in the direction from v_n to v_r at a maximum predefined distance δ_s from v_n. If there is no obstacle between v_n and v, v_n is set as the parent of v. Within a distance δ_r around v, if a better parent exists for v, it is updated. At last, the algorithm undergoes a rewiring process: For neighbor nodes of v within δ_r, if an indirect path through v is shorter than the path to the current parent node, then the parent of this neighbor is set to v. The hyper-parameters of the algorithm are δ_s and δ_r.

Question 13. Propose a data structure for the tree.

Question 14. Propose a method to reconstruct the path from any given node of the tree to the start position.

Question 15. Provide the pseudo-code of your RRT algorithm for path planning.

Question 16. Give the complexity of one iteration of your algorithm.

Question 17. Implement your algorithm and explain your implementation choices.

Question 18. For every scenario given in the test files, provide the path length output by your algorithm, the number of iterations to obtain this path, and the CPU time. Display the path in the environment. Explain how you have chosen the hyper-parameters and the involved tradeoffs.

4.2. Algorithm improvements.

We consider two improvements for the RRT. The first one is *path optimization*. Every path can indeed be optimized based on the triangular inequality.

Question 19. Propose a path optimization method and modify your pseudo-code to take into account this feature. What is the complexity of this method?

Question 20. Test your approach on the proposed scenarios and compare the results to those obtained with the basic algorithm and with PSO (in terms of path length and convergence speed). If you introduce new hyper-parameters, explain how you chose their values and the involved tradeoffs.

The second improvement is *intelligent sampling*. The idea is to sample points close to obstacles vertices or along their periphery rather uniformly random in the environment.

Question 21. Propose an intelligent sampling method and modify your pseudo-code to take into account this feature.

Question 22. Test your approach on the proposed scenarios and compare the results to those obtained with the basic algorithm. Intelligent sampling can be compared and combined with path optimization. If you introduce new hyper-parameters, explain how you chose their values and the involved tradeoffs. What do you conclude on the efficiency of RRT to solve the path planning problem?

5. The Multiple Robot Problem

We now suppose that there are two robots. Around every robot, we define a safe zone, which is a disk centered on the robot and of radius R. To avoid collisions, a robot should not enter the safe zone of the other one. Robots are supposed to move at the same constant speed. They start at $u_s(1)$ and $u_s(2)$ respectively and should reach their respective destinations $u_d(1)$ and $u_d(2)$. Robots should neither hit an obstacle nor collide.

Question 23. Based on the developed methods in the previous sections, propose an algorithm for the 2-robot path planning problem. Explain your proposal, provide the pseudo-code.

Question 24. Implement your algorithm and test your algorithm on the provided scenarios.

6. Final question

Question 25. Explain how you have used AI and github to make your project.

References

[AKYT23] Faten Aljalaud, Heba Kurdi, and Kamal Youcef-Toumi. Bio-inspired multi-uav path planning heuristics: A review. *Mathematics*, 11(10):2356, May 2023.

[GBLV15] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. *Path Planning and Trajectory Planning Algorithms: A General Overview*, pages 3–27. Springer International Publishing, 2015.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4 of *ICNN-95*, pages 1942–1948. IEEE, 1995.

[wik] Particle swarm optimization. `https://en.wikipedia.org/wiki/Particle_swarm_optimization`. Accessed: 2025-10-22.

[XCS+19] Guiping Xu, Quanlong Cui, Xiaohu Shi, Hongwei Ge, Zhi-Hui Zhan, Heow Pueh Lee, Yanchun Liang, Ran Tai, and Chunguo Wu. Particle swarm optimization based on dimensional learning strategy. *Swarm and Evolutionary Computation*, 45:33–51, March 2019.