Path Planning and Execution for an autonomous vehicle, Assignment 1, DD2438 at KTH

GROUP 2
Marco Schouten
1998-05-22
schouten@kth.se

Abstract

The planning and execution of routes for autonomous vehicles is a major research challenge.

The goal of this project is to implement a robust and efficient path planning and navigating system capable of driving a drone and a car as quickly as possible towards a predefined goal in an environment full of obstacles with consistent performances for a variety of maps.

I have carried out simulated experimental tests in a UNITY 3D environment only for the drone. I will compare my findings with those of my course-mates for the drone , and analyze their results for the car model.

1 Introduction (1–2 pages)

To build a navigating system which is robust, efficient and versatile has always been object of intensive research. It is of utmost importance to implement a path planning technique which generally yields stable results for a wide variety of environments. General motion and path planning within an environment rich of obstacles represent one its most popular problems .This is because being able to efficiently and optimally organize the movement of any agent capable of movement has innumerable interests and applications in today's society. For example, organizing the journey for a courier in an optimal way involves enormous social benefits both in terms of customer satisfaction and as a monetary cost as well as pollution. However, it is necessary to specify well what is meant by optimal, in the case of the courier it could be to minimize the distance between departure and arrival, vice versa, for an ambulance that must reach the target in the shortest possible time.

Specifically, this research aims to find end execute a path which minimize the time between departure and arrival towards a specific target position for a Drone and for a Car.

Due to the different nature of the two vehicles, it was reasonable to handle the problem separately for each vehicle. As a result, the performance proved to be better. To find a solution, the problem was divided into two parts: the former to find an optimal path, and the latter to control the vehicle in an optimal way such that it followed the path as closely and as fast as possible.

Among all possible path planning techniques, the Rapidly-exploring random tree (RRT) [4] approach seemed to me the most promising and I developed my solution around that idea. The proposed solution could be improved with a smart sampling heuristic could speed up the algorithm converge towards a solution.

1.1 Contribution

Having worked completely alone, I worked on a navigator solution only for the Drone model provided in the Unity 3D environment.
In this project I have broken down the problem into two separate stages, the first deals with the path planning problem, the second with executing the retrieved path.

The goal of the first stage was to generate an optimal path. To achieve this I implemented a version of RRT added with a heuristic which helped the algorithm to reach faster convergence.

The goal of the second stage was to execute the optimal path i.e. to drive it as fast as possible. To achieve this I chose a Dynamic Point model as the motion model, which is a simplification of the real system. Physics equation determined the acceleration I could put as input to the Drone such that for each consecutive pairs of way-points in the path I could have the appropriate controls.

1.2 Outline

I will explore the state-of-the-art techniques and some of the available literature of motion planning in order to provide an overview of the pros and cons of each strategy in order to choose critically the most suitable method for this specific navigator system Section 2. The tactic chosen for the drone is different from that chosen by my companions for the machine. On top of that I will describe the differences I needed to add specifically for the Drone model in Section 3. In Section 4 I will analyze the results of my findings and compare them with the ones of my course-mates. Lastly, in Section 5 I will address further improvements for the Drone, and discuss about the outcome of this experiments.

2 Related work (1-3 pages)

The first stage is concerned only with Path Planning which is a hard NP problem that may take exponential time to be solved. In this field and a variety of techniques and methods have been developed and studied. Two major challenge arises, firstly the robot does not already have a graph representation of the continuous space it has to navigate, therefore vertices and edges needs to be created beforehand. Secondly an optimal path needs to be found according the built graph. You do not necessarily have to manage those problems separately, in fact it is possible to categorize two different approaches:

- first to discretize, then to search on the graph
- sample and search

A popular discretization techniques that tackles the continuous space is, for example, a Visibility Graph [2]. This technique is based on connecting the vertices of obstacles to each other, in this way the Euclidean distance is minimized as a single step of the path involves the connection between vertices which by construction constitute the minimum distance that a robot must

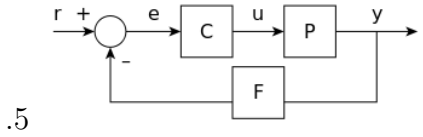
travel to get around these obstacles. However, there are problems with the particular topology of the distribution of obstacles in a map like the resolution for the sampling of the vertices: in areas without obstacles it would be good to have low resolution. Difficulty of generalization led me not to choose this approach. Another strategy to build a graph representing a continuous space filled with obstacles is Voronoi Graph [7]. Its idea is to put way-points between two obstacle, thus minimizing the possibility of collision in order to obtain a path as safe as possible. Though this strategy may be useful for high-risk scenarios or (especially for very large vehicles), for a race it will under-perform compared to other alternatives.

Popular Sample and Search approaches include Rapidly-exploring random trees. RRT is very suitable for managing continuous spaces: the basic idea is to sample points in the space, move the robot a little bit towards that point, if no collision happen, add the reached point and connect it to the closest available node of the graph. Lastly, once the graph is built, a simple backtrack from goal to start will return the list of way-points for the paths. This technique, yet it's simplicity, yields very good paths which tends to be cubic (with fewer turns).

However RRT has some limitations: not only convergence is slow, but also its solutions are far from being optimal. Nevertheless, further improvements can be achieved. In [1], they propose Informed RRT, which restricts the sampling space to an ellipse with foci coinciding with points of departure and arrival. As a consequence the convergence time can be reduced. An other common improvement can be achieved through RRT$*$ [4] which fixes the cubic paths generated by RRT by generating straights paths that are ideal to reach the goal with high speed. Additionally, RRT$*$ claims to converge towards the optimal solution, but theoretically, an infinite amount of time is required. To correct this problem, state-of-the-art Smart-RRT$*$ [3] add a smart sampling and path optimization to aim for faster convergence towards an optimal or close to optimal solution provided by the base RRT$*$.

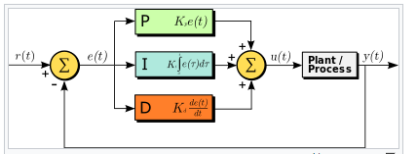
Lastly, a very popular approach was Hybrid A$*$ [5], an extension of A$*$ for continuous spaces. In a nutshell it allows an object such as a car to respect its non-holonomic constraints, which prevent it from rotating on itself. This modeling has made this strategy the most used to obtain a smoothly passable path from the car. As expected it was the best performing path planning algorithm for the car.

The second stage is concerned finding optimal controls such that the vehicle can execute the path as closely and as fast as possible. In the field of control theory, the most popular and successful approach to ensure stability is to implement a control loop with feedback. A PID (pro-



.5

Figure 1: closed-loop regulator for a given plant P



.5

Figure 2: PID controller

portional–integral–derivative) controller or a single Closed Loop Regulator proved being very successful techniques in finding appropriate controls, as for example [6].

3 Proposed method (1-4 pages)

I have implemented a solution only for the drone. The solution is structured in two stages: first finding a path, then executing it. The big structure of my solution refers to the following sequence of steps below:

1. Finding a Path.
2. Converting a Path to Controls
3. Controlling the drone

A Unity 3D object "Drone" needed a script that regulated its behavior, therefore code was added to a Drone-AI script. The script has two major functions which are called by the UNITY 3D software at different times:

- Start()
- FixedUpdate()

The Start function has code that runs to only once at the beginning and serves to initialize all important variables. Code in this section could is run before the beginning of the race, and therefore it does not have hard time constraint. In this section I have implemented an algorithm capable of finding the path, then to translate the path into controls suitable for the Drone.

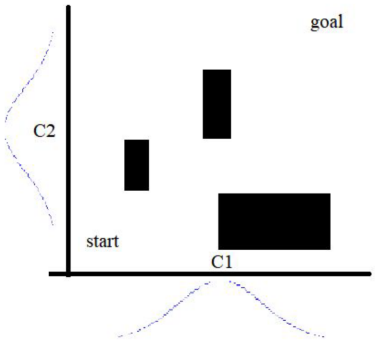


Figure 3:

Whereas the FixedUpdate function is called every frame, and I used that to scan the controls from the list and give the acceleration to the drone at the right time.

3.1 Finding a Path

As mentioned in the introduction, my implementation is based on Rapidly-exploring random trees (RRT). First and foremost, I built an efficient data-structure capable of storing and access nodes. A key part of an RRT based algorithm is to add a new node to the already existing graph and the most crucial step is to link it with the closest node. This is one of the most costly operations and it needs to be performed countless times. Basically, one needs to scan the whole graph and compute an euclidean distance between each node so that the minimum can be found. On top of that one needed to check whether a point can be added to the drone, that is it lies in a free-space with a certain distance from a thresholds, this is checked by a function that inspects the Traversability Matrix.

Next, an efficient heuristic should inform and influence the sampling to reach a faster convergence. The idea is to have a smaller probability of sampling at the corners of the map, as usually it is most likely that such areas are not to be run across. A Gaussian probability density function with very large variance and mean located in somewhere in between the start and goal position should serve for this purpose [Figure 3].

To keep track of the edges of this graph, I have used a List which uses as indexes the position of a node and as values the index of its parent. Therefore when the algorithm,by adding nodes without collision, has reached a close distance to the goal, one can efficiently retrieve the reverse-ordered path just by rereading an element of the list as index for retrieved the subsequent node.

Lastly to get the correct order of way-point, one need to reverse the order of the list. Algorithm 1 summarizes the scheme of the before-mentioned steps.

Algorithm 1: Path Planning, Start()

Result: Path as a List of way-points
Graph INIT;
iterations INIT;
threshold ;
q_{goal} INIT;
while *While* $i \leq iterations \& goal\_distance \leq threshold$ **do**
  $q_{rand} \leftarrow$ RANDCONF(q_{goal}) ;
  $q_{near} \leftarrow$ NEAREST(q_{rand}) ;
  $q_{new} \leftarrow MOVE(q_{near}, q_{rand})$;
  if *isTraversable*(q_{new}) **then**
    ADD q_{new};
  end
  goal_distance $\leftarrow$ DIST(q_{goal});
  i++;
end
path $\leftarrow$BACKTRACK(G);
APPEND(path, q_{goal}) ;

3.2 Converting a Path to Controls

To convert a path to controls I have set a fixed time the vehicle has in order to move from a way-point to the next. The goal is to control the drone such that it moves from way-point to way-point in a straight line. Though I am aware that it is not optimal to have a velocity equal to 0 at the end of each way point, it should serve as a solution given the time constraints of this project, moreover, by including some pruning in the graph, it could still provide acceptable results.

The idea is that I have to different accelerations, one positive (a_1), and the other one to stop the vehicle (a_2) as in Figure4.

to achieve this, one can scan the whole path, get the current position and computing the two components of the acceleration. Here-under "next" and "actual" refer respectively to the way-points in the path:

$$x_{dir} = x_{next} - x_{actual} \tag{1}$$

$$z_{dir} = z_{next} - z_{actual} \tag{2}$$

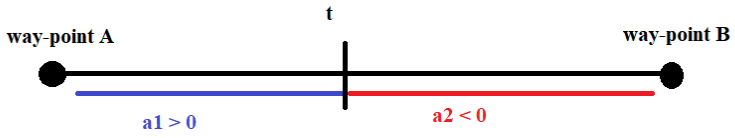


Figure 4:

Secondly, you can compute the accelerations that you give as input to the node.

$$\begin{cases} a_{1,x} = x_{dir}/(t^2) \\ a_{1,z} = z_{dir}/(t^2) \\ a_{2,x} = (x_{dir}/2 - a_{1,x} * t) * 2/(t^2) \\ a_{2,z} = (z_{dir}/2 - a_{1,z} * t) * 2/(t^2) \end{cases} \tag{3}$$

3.3 Controlling the Drone

Lastly, it is very important to give the acceleration to the drone at the right times. The FixedUpdate function is called every frame. Unity3D API "Time.fixedDeltaTime" can tell me how long does a frame last, therefore I know how much time passes by adding that amount to a variable with global scope that serves as a counter. Additionaly it is needed to a "time parameter" with the same value as the one set before ("t" in the formulas (3)) that dictates the amount of time needed for the vehicle to move from a way-point to another. The idea is that you repeat the acceleration until enough time as passed, you can perform check when it is time to move on on the control as descripted in the algorithm before.

4 Experimental results (1-4 pages)

In this section I will comment about the results of my course-mates first and then address the results I got in the experimental setup.

The outcome containing the results of all groups can be seen in Table 5. Group 3 had the best results and was the Drone "specialist". Their idea was to make the drone decelerate when in proximity of a way-point. They had a certain threshold to check that. Moreover having the time to be dependent on the distance of the node, was also a good idea. For the car they also took into account the sharpness of the curve. Group 7 was the car "specialist"

Algorithm 2: Giving Controls to the Drone

Result: void
t_{actual} INIT;
t_{frame} INIT;
t INIT ;
controls INIT;
while *FIXED UPDATE ()* **do**
  if $t_{actual} < (current\_step + 1) * t$ **then**
    MOVE(controls[current_step].x, controls[current_step].z ;
  else
    current_step++ ;
  end
  $t_{actual} \leftarrow t_{actual} + t_{frame}$;
end

Add your results below (replace 150 with actual value)

Leader Board			CAR MODEL						DRONE MODEL						Total time		Leader Board	
			TerrainA	TerrainB	TerrainC	Terrain D	Terrain E		TerrainA	TerrainB	TerrainC	Terrain D	Terrain E					
G3	381.5	G1	56.02	88.29	14.2	46.37	55.01	G1	44.42	117.26	10.02	39.66	58.39	G1	529.6	G3	381.5	
G9	385.2	G2	150	150	150	150	150	G2	150	150	150	150	150	G2	1500.0	G9	385.2	
G7	420.6	G3	34	80	10.13	39.12	37.24	G3	36.56	66.36	10.3	30.35	37.43	G3	381.5	G7	420.6	
G13	503.7	G4	57.94	75.58	10.02	160	53.56	G4	56.7	109.5	11.36	59.12	73.68	G4	667.5	G13	503.7	
G1	529.6	G5	90	150	17	150	110.55	G5	150	150	150	150	150	G5	1267.6	G1	529.6	
G10	585.5	G6	66	150	21	150	150	G6	150	150	150	150	150	G6	1287.0	G10	585.5	
G4	667.5	G7	25.01	52.1	7.3	28.1	28.2	G7	55.75	90.7	15.67	55.96	61.85	G7	420.6	G4	667.5	
G14	702.0	G8	150	150	150	150	150	G8	150	150	150	150	150	G8	1500.0	G14	702.0	
G15	897.3	G9	27	55	8.3	34.1	33.9	G9	46	84	10.1	37.7	49.1	G9	385.2	G15	897.3	
G12	1018.0	G10	33.28	126.59	7.42	66.32	68	G10	59.38	101.43	10.38	48.62	64.12	G10	585.5	G12	1018.0	
G5	1267.6	G11	150	150	78	150	150	G11	150	150	150	150	150	G11	1428.0	G5	1267.6	
G6	1287.0	G12	150	150	150	150	150	G12	61	98	12	46	51	G12	1018.0	G6	1287.0	
G11	1428.0	G13	59.3	92	15.3	42.78	53.96	G13	45.2	90	11.28	42.38	51.5	G13	503.7	G11	1428.0	
G2	1500.0	G14	63	89	16	58	64	G14	82	137	30	84	79	G14	702.0	G2	1500.0	
G8	1500.0	G15	150	150	9.12	110	150	G15	65	125	10	58.06	70.09	G15	897.3	G8	1500.0	
G16	1500.0	G16	150	150	150	150	150	G16	150	150	150	150	150	G16	1500.0	G16	1500.0	
		Best time	25.01	52.1	7.3	28.1	28.2	Best time	36.56	66.36	10	30.35	37.43	Best time	381.5			
		Best Group	G7	G7	G7	G7	G7	Best Group	G3	G3	G15	G3	G3	Best Gro	G3			

Figure 5: Overall Results

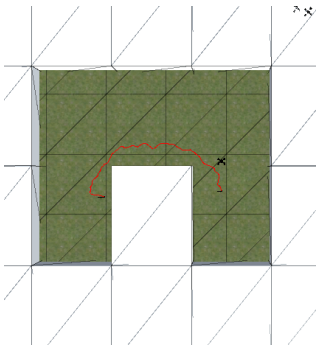


Figure 6: Planned Path

meaning that they obtained the best results for the car. The reason being is that first they optimized their code for the Car, and later on they adapted it to the drone. They discarded RRT due to its high computational demand and for their final implementation combined an expanded visibility graph with A∗. A very good idea of them was to use a simpler PID controller as for smaller angles they could make use of approximations in their formulation to compute the angular velocity. They also transformed the plane such that the goal position was the center and your current position tells you how much the vehicle has deviated from the path. As a result they could perform very good steering for the car.

4.1 Experimental setup

For the path planning I managed computing good collision free paths as shown in figure6.

Unfortunately, though I had a path, I had hardships controlling the drone and making it follow it closely.

4.2 Analysis of Outcome

Hybrid A∗ was the best approach for the car because it could provide a more detailed model dealing with the steering, which was a critical constraint for the car.

RRT∗ was the best strategy for the drone because it yielded straight paths with fewer turns. As a consequence, drone could turn in less space and have a shorter path as they had no constraints for steering.

4.3 Additional sub-section:

How much time you spent on this assignment, and risk management

- Week 3 was introductory week. I put effort trying to reach out my first lab partner Mikaela Funkquist, the goal was to read about c# and organizing.
- week 4 we had the lecture about path planning, so I spent time reviewing literature, learning c#, watching tutorials for unity as everything was completely new to me. I was assigned a new lab partner Zhenlin Zhou. So I had to re-schedule everything according to the member's preferences and spent time organizing. I managed to have the first (and only) meeting near the end of the week, but still there was not any evident red flags as I never met my partner before. As I was team leader I took initiative of writing the slides and motivating RRT over other strategies. though we had a slower start, I was optimistic, as I did not know my partner and there was still time.
- week 5 since my partner didn't answer me, I preferred to wait for a few days, but eventually I decided to report the case and I tried to cram the group work of week 5 all alone over the last few days.
- week 6 I Was working alone full-time on the drone model, and I spend lots of time debugging UNITY and trying to find documentation.

Why you thought that your solution would work I had a working planned path, so the only problem remained to be addressed was to control the drone. I approximated the drone model and calculated all the accelerations.

Why your solution did not work I firmly believe that there is something wrong with the Time.fixedUpdate and the FixedUpdate functions call. The reason is that my drone does not follow the path, though it starts in the right direction, and though the accelerations are alright, the drone does not move accordingly. There is an inconsistency between the computed accelerations and the movement of the drone. The reason of this inconsistency probably lies in the physical units or in the duration time of any acceleration. All the computed accelerations are in m/s^2 and the distance unit in the map is assumed as 1m.

How you made the risk assessments in weeks 4,5,6. week 4 was optimistic as I did not know my partner yet. Week 5 was already late, and I was concerned about not completing the project within the deadline. I tried

to put extra effort, but unfortunately the Time bug couldn't make me hand in a solution for week 6.

5 Summary and Conclusions (0.5-1 page)

Ad-hoc solutions yield better results than a one-fit-all solution. On average the most successful and used algorithm were Hybrid A$*$ for the car and RRT$*$ for the drone. Reason being that the different models had different constraints and therefore, as they behaved differently, they were more agile moving towards a certain type of path. To sum up both approaches have pros and cons for a particular environment. As there was not a path planning strategy clearly better than others, no holy grail of path planning could be found. Further improvements could be achieved by tuning parameters, checking other heuristics: testing different combinations of path planning strategies.

References

[1] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[2] Han-Pang Huang and Shu-Yun Chung. Dynamic visibility graph for path planning. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2813–2818. IEEE, 2004.

[3] Fahad Islam, Jauwairia Nasir, Usman Malik, Yasar Ayaz, and Osman Hasan. Rrt-smart: Rapid convergence implementation of rrt towards optimal solution. In *2012 IEEE international conference on mechatronics and automation*, pages 1651–1656. IEEE, 2012.

[4] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[5] Janko Petereit, Thomas Emter, Christian W Frey, Thomas Kopfstedt, and Andreas Beutel. Application of hybrid a* to an autonomous mobile robot for path planning in unstructured outdoor environments. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6. VDE, 2012.

[6] Atheer L Salih, M Moghavvemi, Haider AF Mohamed, and Khalaf Sallom Gaeid. Modelling and pid controller design for a quadrotor unmanned air vehicle. In *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, volume 1, pages 1–5. IEEE, 2010.

[7] Jiankun Wang and Max Q-H Meng. Optimal path planning using generalized voronoi graph and multiple potential functions. *IEEE Transactions on Industrial Electronics*, 67(12):10621–10630, 2020.