Defence Science Journal, Vol. 67, No. 1, January 2017, pp. 45-49, DOI : 10.14429/dsj.67.10295

Goal Directed Approach to Autonomous Motion Planning for Unmanned Vehicles

David Boon Moses E.* and G. Anitha

Department of Aerospace Engineering, Madras Institute of Technology, Anna University, Chennai – 600 044, India
**E-mail: boonflies@gmail.com*

ABSTRACT

Advancement in the field of autonomous motion planning has enabled the realisation of fully autonomous unmanned vehicles. Sampling based motion planning algorithms have shown promising prospects in generating fast, effective and practical solutions to different motion planning problems in unmanned vehicles for both civilian and military applications. But the goal bias introduced by heuristic probability shaping to generate faster solution may result in local collisions. A simple, real-time method is proposed for goal direction by preferential selection of a state from a sampled pair of random state, based on the distance to goal. This limits the graph motions resulting in smaller data structure, making the algorithm optimised for time and solution length. This would enable unmanned vehicles to take shorter paths and avoid collisions in obstacle rich environment. The approach is analysed on a sampling based algorithm, rapidly-exploring random tree (RRT) which computes motion plans under constrain of time. This paper proposes an algorithm called 'goal directed RRT (GRRT)' building on the basic RRT algorithm, providing an alternative to probabilistic goal biasing, thereby avoiding local collision. The approach is evaluated by benchmarking it with RRT algorithm for kinematic car, dynamic car and a quadrotor and the results show improvements in length of the motion plans and the time of computing.

Keywords: Path planning; Unmanned vehicles; Quadrotor; Sampling-based motion planning; RRT

1. INTRODUCTION

Different approaches have been proposed to realise planning in dynamic environments[1]. Sampling[2] based motion planning algorithms have made real time planning of motion for unmanned vehicles possible[3]. A bias towards the goal[4] is usually added by giving an increased probability to the goal states which tends to increase the rate of convergence of the planner. Shaping the probability distribution function[5] which is used to generate random samples to grow a tree has shown improvements in the rate of convergence of the path planner[6]. The negative side of increasing the bias is that like randomised potential field it becomes too greedy also leading to local minima problem[7]. The goal directed approach presented in this paper is intended to provide an indirect goal bias without the probability based goal bias included in sampling algorithms. With the known advantage of increase in the rate of convergence, this approach tends to provide a solution with improved solution length. Study shows the improvement of goal directed approach over the probabilistically goal biased RRT algorithm for different unmanned vehicle platforms[8].

2. RAPIDLY EXPLORING RANDOM TREES

Path planning involves defining a state space, X which constitutes both free region, X_{free} where the path can proceed and obstacle region, X_{obs} where the path cannot proceed through. The state space enables representation of both configuration and velocity in a space. A collision detection algorithm enables to determine if a state, x causes collision in the modeled world. Thus selecting a number of collision-free states and computing a path from the start, x_{init} to the goal state, x_{goal}.

The RRT algorithm[9] considers dynamic constraints for path planning.

Algorithm 1: BUILD RRT(x_{init})

a. $T.init(x_{init})$
b. **for** $k = 1$ **to** K **do**
c. $x_{rand} \leftarrow RANDOM\_STATE()$;
d. EXTEND(T, x_{rand});
e. *Return* T

Given an initial and goal state, a tree is constructed with the root of the tree in the initial state and branching till the goal state is reached. Adding nodes to the tree involves generation of random state x_{rand}, within the state space using the *RANDOM_STATE* function. The selection of random nodes can be controlled by introducing a bias towards goal using probabilistic shaping in *RANDOM_STATE* function. Followed by the generation of a state x_{rand}, extension of the tree is realised by the *EXTEND* function.

The node in the tree which is nearest to the random state is identified using *NEAREST_NEIGHBOR* function. The function uses a certain metric to compute the distances of each node of the tree with the random state. The node with the shortest distance to the random node, called nearest neighbour, x_{near} is

Received : 06 July 2016, Revised : 01 December 2016
Accepted : 02 December 2016, Online published : 22 December 2016

selected to connect the random node to the tree.

Algorithm 2: RRT EXTEND(T, x)

a. $x_{near} \leftarrow NEAREST\_NEIGHBOR(x,$ T);
b. **if** *NEW_STATE*(x, x_{near}, x_{new}, u_{new}) **then**
c. T .*add vertex*(x_{new})
d. T .*add edge*(x_{near}, x_{new}, u_{new})
e. **if** $x_{new}=x$ **then**
f. *Return Reached*;
g. **else**
h. *Return Advanced*;
i. *Return Trapped*;

NEW_STATE function is to make a motion towards x_{rand} from x_{near} using an input u_{new} or an incremental distance, ε. The function checks if the random state, x_{rand} and also the intermediate states from x_{near} are reachable. If the random sample state x_{rand} was reached then the function returns *Reached*. If the tree only progresses towards x_{rand} then it returns *Advanced* and if the motion is trapped by an obstacle it returns *Trapped*. The new state is represented as x_{new}.

3. GOAL DIRECTED RRT PATH PLANNER

A different variations[10] has been proposed in this paper to the basic RRT algorithm. The current existing variations to the algorithm are classified under three categories as methods for generating random state, methods to extend towards a new node and methods to add bias to the goal. Every type of variation has its own advantages. The goal directed RRT path planner introduces a new method of adding bias towards the goal primarily by using a *NEAREST_TO_GOAL* function resulting in faster rate of convergence of the planner. And for certain reasons the experiments also show a reduction in solution length.

Algorithm 3: BUILD DIRECTED_RRT(x_{init})

a. T .*init*(x_{init})
b. **for** k = 1 **to***K***do**
c. $x_{rand1} \leftarrow RANDOM\_STATE()$;
d. $x_{rand2} \leftarrow RANDOM\_STATE()$;
e. $x_{rand} \leftarrow NEAREST\_TO\_GOAL(x_{rand1}, x_{rand2})$;
f. *EXTEND*(T , x_{rand});
g. *Return* T

With the knowledge of initial and goal state, the planner starts by generating two random states x_{rand1} and x_{rand2} instead of one random state generated by the basic RRT planner in the free region using the *RANDOM_STATE* function. The difference in the *RANDOM_STATE* function introduced in GRRT is that it does not include probabilistic shaping towards goal as in RRT and hence avoiding the local minima problem.

The GRRT achieves the search towards the goal by the introduction of *NEAREST_TO_GOAL* function which selects the random state closer to the goal out of the two random states, x_{rand1} and x_{rand2}. The randomstate out of x_{rand1} and x_{rand2} which is closest to the goal becomes the chosen random state x_{rand} for further growth of the tree.

The number of random states to choose from was set to two after simulation results showed best performance when the number of random states was limited to two. So, GRRT algorithm is proposed to include a preferential selection out of just two random states.

The *EXTEND* function in GRRT carries out the same functionality as in RRT by extending an edge from the tree towards the chosen x_{rand}. The *NEAREST_NEIGHBOUR* function then computes the neighbour in the tree x_{near}, closest to the chosen random state, $x_{rand.}$ Then the tree extends towards the random state. If the tree reaches x_{rand}, the function returns *Reached* or if it advances but does not reach x_{rand} it returns *Advanced* and if the motion gets trapped by an obstacle, it returns *Trapped.*

The RRT tree structure exhibits the advantage of exploration of the entire search space. The tree structure of Goal directed RRT has an inclination towards the goal. This initiates a faster rate of convergence of the Goal directed RRT algorithm.

A possible variation to the proposed approach is if x_{rand1} or x_{rand2} lies in $X_{obs,}$ whichever lies in X_{free} can be selected within the *NEAREST_TO_GOAL* function.

4. EXPERIMENTAL SETUP

The experiments were carried out on Intel Core i3 CPU with 2.53 GHz processor with 4GB RAM. The OMPL[11] motion planning library was used for benchmarking the proposed algorithm. OMPL has provision for carrying out benchmarking tests by including new motion planning algorithms. The GRRT was included in OMPL for benchmarking the algorithm. OMPL includes different environment wherein the benchmarking can be carried out. Scripts available with OMPL to visualise the benchmarking results can be used for data visualisation.

On an average 25 runs were made for each planner in every scenario with a minimum planning time of 10 seconds to a maximum of 100 seconds. Whenever goal bias i.e., adding goal state to the distribution with a higher probability was considered, it was set to 0.05.

Algorithm 4: DIRECTED_RRT EXTEND(T, x)

a. $x_{near} \leftarrow NEAREST\_NEIGHBOR(x,$ T)
b. **if** *NEW_STATE*(x, x_{near}, x_{new}, u_{new}) **then**
c. *T.add_vertex*(x_{new})
d. *T.add_edge*(x_{near}, x_{new}, u_{new})
e. ***if*** $x_{new} = x$ ***then***
f. *Return Reached*
g. ***else***
h. *Return Advanced*
i. Return Trapped

4.1 Kinematic Car Planning

The configuration of a car can be expressed as,

$$q = (x, y, \theta) \in R \times S^1 \quad (1)$$

where the position is,

$$(x, y) \in R^2 \quad (2)$$

and the orientation is,

$$\theta \in S^1 \quad (3)$$

The definition of the benchmarking parameters is as follows:

Graph Motion: The number of edges in the constructed graph.

Solution Length: The length of the found solution.

Solution Segments: The number of segments on the solution path.

Time: The amount of time spent planning, in seconds.

Solution Difference: If the solution is approximate, this refers to the distance of goal from the end-point of the found approximate solution.

Solved: Indicating whether the planner found a solution (even approximate solution is considered)

The benchmarking was carried out on maze environment as shown in Fig. 1. As seen from Table 1 and Table 2, GRRT shows a decrease in graph motion, which represents a reduced graph tree data structure resulting in a faster search. Any graph search algorithm would be benefited by the reduced graph data structure.

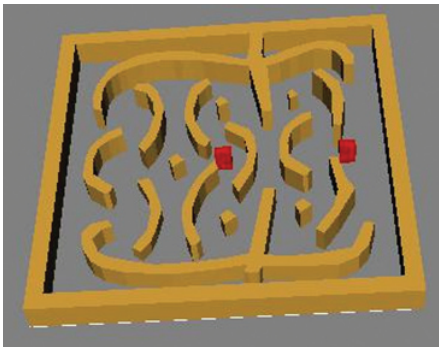

Figure 1. Maze environment for kinematic car planning.

Table 1. Kinematic car benchmarking results

Performance measure	RRT	GRRT
Graph motions	42057.32	32661.72
Solution length	24.278	24.410
Solution segments	18.16	16.32
Time	7.93	6.85
Solved	8	12

Table 2. Percentage change for kinematic car

Performance measure	Improvement - GRRT (%)
Graph motions	decrease of 22.34
Solution segments	decrease of 10.13
Time	decrease of 13.62
Solved	increase of 50

4.2 Dynamic Car Planning

Dynamic car planning involves considering the acceleration $\ddot{q}$ in addition to velocity $\dot{q}$ and configuration q.

For dynamic car planning, the state is,

$$s = (x, y, \theta, s, \varphi) \tag{4}$$

where the position,

$$(x, y) \in R^2 \tag{5}$$

the orientation,

$$\theta \in S^1 \tag{6}$$

the translational velocity,

$$s \in R \tag{7}$$

the steering angle,

$$\varphi \in R \tag{8}$$

The benchmarking of algorithms for a dynamic car was performed in 'Random Polygon' environment shown in Fig. 2, with a maximum runtime of 10 s for 25 runs for each planner. Similar to kinematic car benchmarking, different benchmarking parameters like solution length, time, solution segments, graph motions were considered for benchmarking. The results for a dynamic car are shown in Table 3 and the percentage change in Table 4.

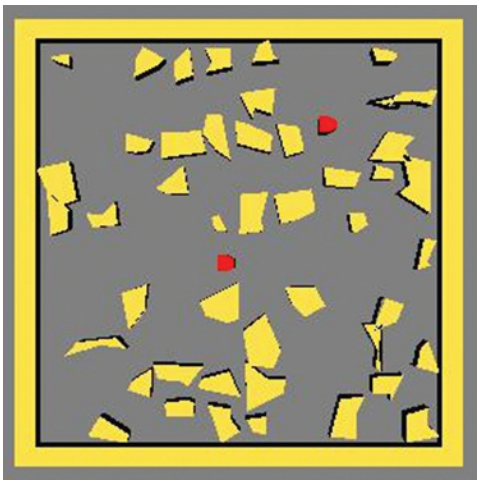

Figure 2. Random polygon environment for dynamic car planning.

Table 3. Dynamic car benchmarking results

Performance measure	RRT	GRRT
Graph motions	17841.4	17591
Solution length	83.915	76.8071
Solution segments	56.4	50.24
Solution difference	0.96536	0.770209
Time	9.86	9.33
Solved	2	4

Table 4. Percentage change for dynamic car

Performance measure	Improvement – GRRT (%)
Graph motions	decrease of 8.47
Solution length	decrease of 10.92
Solution segments	decrease 20.21
Time	decrease of 5.38
Solved	increase of 100

4.3 Quadrotor Planning

Quadrotor motion planning[12] is a very hard motion planning problem[13] even with a simplified dynamics model. Quadrotor motion using predefined waypoints has become a basic capability but generating motion plans autonomously is still a challenging research area. The dynamics of the quadrotor are defined as follows:

$$m\ddot{p} = u_0 n - \beta\dot{p} - mg \tag{9}$$

$$\alpha \in (u_1, u_2, u_3)^T \tag{10}$$

where p is the position, n is the Z-axis of the body frame in world coordinates, α is the angular acceleration, m is the mass and β is the damping.

The system is controlled through,

$$u = (u_0, u_1, u_2, u_3) \quad (11)$$

The control inputs include u_0, which is the mass normalised collective thrust, the roll rate about the body axes u_1, the pitch rate u_2 and the yaw rate u_3.

The quadrotor benchmark was carried out in 'Cubicles' environment as shown in Fig. 3, with a maximum time limit for every planner execution set to 10 s for one trial and to 100s for another trial.

As seen in Table 5, the goal direction approach defined in this paper improves on the solution length. Even in a high dimensional environment like quadrotor planning, the directed approach presents itself as a valuable addition to the RRT algorithm. The improvement in solution becomes significant as time of planning increases and so would be profited by higher computing power.

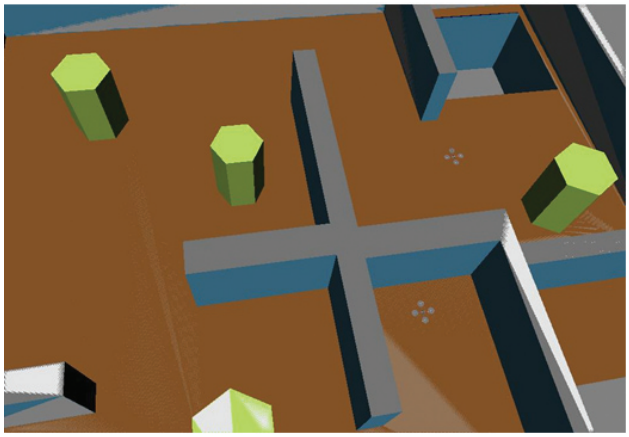

Figure 3. Cubicles environment for quadrotor planning.

Table 5. Percentage change for quadrotor

Performance measure	Improvement GRRT	
	10s (per cent)	100s (per cent)
Solution length	↓ 7.6	↓ 14.5
Solution segments	↓ 4.4	↓ 5.2

5. WORK IN PROGRESS

Currently, work is being carried out to implement GRRT algorithm in a real world environment with quadrotor as an unmanned vehicle platform. This serves the cause of furthering the research on autonomous quadrotor planning as well as to analyse the proposed algorithm.

6. CONCLUSION

In this paper we have presented the first results for an algorithm which is proposed to overcome the local collisions caused by goal bias in unmanned vehicles along with improving the solution length and computing time which is required to perform real-time planning. The GRRT algorithm presents a faster solution with shorter solution length and would be useful in the case of large search spaces and in places where planning is to be accomplished in previously unknown search spaces. The goal bias is introduced by a modified approach in the selection of sampled random state. The test results of Goal directed RRT algorithm shows significant improvements in time and solution lengths. The algorithm was formulated in order to overcome the disadvantages of heuristic probability shaping. This goal directed approach can be considered for different types of sampling based planners. The effectiveness of this method will be determined with experiments on unmanned vehicles in unknown environment.

REFERENCES

1. Daniel-Ioan, Curiac & Constantin, Volosencu, Path planning algorithm based on Arnold cat map for surveillance UAVs. *Def. Sci. J.*, 2015, **65**(6), 483-488. doi: 10.14429/dsj.65.8483
2. Karaman, S. & Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Res.*, 2011, **30**(7), 846-894. doi: 10.1177/0278364911406761
3. Singh, G.K. & Gopal, A. Path planning in the presence of dynamically moving obstacles with uncertainty. *Def. Sci. J.*, 2010, **60**(1), 55-60.doi: 10.14429/dsj.60.107
4. LaValle, S.M. & Kuffner, J.J. Randomized kinodynamic planning. *Int. J. Robotics Res.*, 2001, **20**(5), 378-400. doi: 10.1177/02783640122067453
5. Akgun, B. & Stilman, M. Sampling heuristics for optimal motion planning in high dimensions. *In* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011. doi: 10.1109/IROS.2011.6095077
6. Urmson, C. & Simmons, Reid. Approaches for heuristically biasing RRT growth. *In* Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. doi: 10.1109/IROS.2003.1248805
7. LaValle, S.M. & Kuffner, J.J. Rapidly-exploring random trees: progress and prospects. *In* Workshop on Algorithmic Foundations of Robotics, 2000.
8. Kuwata, Y.; Fiore, G.A.; Teo, J.; Frazzoli, E. & How, J.P. Motion planning for urban driving using RRT. *In* IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2008. doi: 10.1109/IROS.2008.4651075
9. LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, 1998.
10. Kuffner, JJ. & LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. *In* Proceedings of IEEE International Conference on Robotics and Automation, 2000, pp. 995-1001. doi: 10.1109/ROBOT.2000.844730
11. Sucan, I.A.; Mark Moll & Kavraki, L.E. The open motion planning library. *IEEE Robotics Automation Magazine*, 2012, **19**(4), 72-82. doi: 10.1109/MRA.2012.2205651
12. Allen, R. & Pavone, M. A real-time framework for kino dynamic planning with application to quadrotor obstacle avoidance. *In* AIAA Conference on Guidance, Navigation and Control, San Diego, CA, 2016. doi: 10.2514/6.2016-1374
13. Richter, C.; Bry, A. & Roy, N. Poly-nominal trajectory planning for aggressive quadrotor flight in dense indoor environments. *In* Proceedings of the International Symposium of Robotics Research, 2013. doi: 10.1007/978-3-319-28872-7_37

CONTRIBUTORS

Mr David Boon Moses E. obtained his ME (Avionics) from Madras Institute of Technology (MIT), Anna University. Currently pursuing his PhD (Avionics) at MIT, Anna University. He is working in the area of autonomous UAV motion planning. His areas of interest include robotic simulation and motion planning.
In the current study, he conceived the algorithm, performed simulation and benchmarked the results.

Dr G. Anitha obtained her MTech (Control & Instrumentation) from IIT, Madras and PhD from Madras Institute of Technology, Anna University. Presently, she is working as Assistant Professor in the Department of Aerospace Engineering, MIT campus, Anna University, Chennai. Her areas of interest include avionics system, navigation guidance and control, and image processing.
In the current study, she was involved in conception of the algorithm along with defining the framework for simulation.