A Suboptimal Path Planning Algorithm Using Rapidly-exploring Random Trees

Mangal Kothari[a,1], Ian Postlethwaite[b], Da-Wei Gu[a]
[a]Control Research Group, University of Leicester, UK
[b]Northumbria University, UK

Abstract
This paper presents path planning algorithms using Rapidly-exploring Random Trees (RRTs) to generate paths for unmanned air vehicles (UAVs) in real time, given a starting location and a goal location in the presence of both static and pop-up obstacles. Generating near optimal paths in obstacle rich environments within a given short time window is a challenging task. Hence we first generate a path quickly using RRT by taking the kinematic constraints of the UAV into account. Then we develop an anytime algorithm that yields paths whose quality improves as computation time increases. When the UAV detects a pop-up obstacle the path planner re-generates a new path from its current location. In order to track a generated path, an effective guidance law with a switching mechanism based on pursuit and line of sight guidance laws is developed. Simulation studies are carried out to demonstrate the performance of the proposed algorithm.

Key words: Path Planning Algorithm, RRTs, Pursuit guidance, LOS guidance

1. INTRODUCTION

Recently, unmanned aerial vehicles (UAVs) have been deployed for various search and surveillance missions. One of the main objectives of a mission is to generate a path that a UAV can follow. A path is usually described in terms of a set of waypoints that the UAV has to visit sequentially. The path planning problem is to generate a path in terms of waypoints taking various environmental and physical constraints of the UAV into account. Designing a path planner in an obstacle rich environment that contains static and pop-up obstacles is difficult. The difficulty further increases when a near optimal solution is sought in real time while accounting for kinematic constraints of the UAV.

For UAV applications, where the UAV has to fly in obstacle rich environments, the computational complexity of the path planning algorithm is an important issue that needs to be addressed. Since, the UAV is in motion at high velocity, the path planner has to find a path quickly. When unknown obstacles are detected, a new path needs to be generated in real time from its current location to avoid the obstacles. If the path planner fails to generate a feasible solution within a given time window, the UAV may collide with the obstacles leading to mission failure.

The path planning problem is an important area of interest in the field of UAVs and robotics with requirements on optimality, completeness and computational complexity. A number of path planning algorithms have been developed in these fields such as road map, cell decomposition, potential field, etc [1]–[2]. However, few of these try to solve the planning problem in its full generality [1]. Depending on the nature of the problem, some techniques work better than others. An extensive introduction to the path planning problem and existing solutions may be found in [1]–[3]. The computational time of deterministic and complete algorithms grows exponentially with the dimension of the configuration space. Therefore these algorithms cannot be employed for real time UAV path planning problems, specially those problems that contain rich static and pop-up obstacles.

Recently, sampling based motion planning has gained much interest as an alternative to complete motion planning algorithms [4]–[5]. For example, Rapidly-exploring Random Trees (RRTs) have been shown to be effective in UAV applications [6]–[7]. The standard RRT algorithm produces

Email addresses: mk221@le.ac.uk (Mangal Kothari), ian.postlethwaite@northumbria.ac.uk (Ian Postlethwaite), dag@le.ac.ukn (Da-Wei Gu)

[1]Corresponding author

a time parameterized set of control inputs to move from a starting position to a goal configuration. The performance of the path planner is dependent on the accuracy of the state model being used. Moreover, sensor inaccuracies, wind disturbance and unmodeled dynamics affect the performance of the path planner. Hence, open loop path planners cannot provide the desired performance [7]. To address this issue, RRTs are extended in the output space, where the search is carried out to determine waypoints [7]. Then, a robust controller is employed to track the generated path of waypoints. It is assumed that the controller keeps the UAV on the waypoints' path (waypath) while rejecting disturbances. Although RRT is an effective and computationally efficient tool for complex online motion planning, the solution is far from optimal due to its random exploration of the search space.

In this paper, we incorporate actual trajectory characteristics in the algorithm based on the kinematic model of the UAV, to avoid possible collisions. Then a suboptimal approach combining actual trajectory information characteristics with an anytime approach [8] is developed to deal with static and pop-up obstacles in real time. Moreover, this approach renders a suboptimal solution by continuously improving the quality of the solution. In order to track the generated path precisely, a robust guidance law with a switching mechanism based on pursuit and line of sight guidance laws is developed.

The rest of paper is organized as follows. Section II describes the path planning problem and scenario. In Section III, a suboptimal algorithm for path planing is developed. In Section IV, in order to track a generated path, an elegant guidance law is developed. In Section V, the performance of the path planner and guidance law are shown by simulation results. Finally, in Section VI, concluding remarks are presented.

2. PROBLEM STATEMENT

A mission is considered where a UAV needs to travel from a starting location to goal location through an environment consisting of static and pop-up obstacles. We assume full knowledge of the terrain map, however, pop-up obstacles may encounter during the flight. The problem becomes difficult because the motion constraints of UAVs (e.g. turn radius) are comparable with the environment. This necessitates that UAV constraints must be taken into account while finding a path so that unexpected collisions can be avoided. The objective here is to determine a path which is near optimal in real time while avoiding any known a priori and pop-up obstacles and while satisfying physical constraints. It is assumed that the UAV has constant velocity and limited sensor range. Since the UAV has limited sensor range, it can detect the pop-up obstacles when they are in the sensing range. If the obstacles are on the flight path, the UAV has to re-plan its path from its current position. Determining a solution which satisfies physical constraints while avoiding pop-up obstacles in the environment is difficult and computationally intensive. Hence, incorporation of actual trajectory characteristics strategy with anytime approach is employed to generate a time dependent improvised path.

3. A SUBOPTIMAL ALGORITHM

In this section, we shall first discuss the standard RRT algorithm and its variants. Then, we present a greedy strategy which not only enhances the computational capability but also reduces unnecessary path segments. Finally, combining the greedy strategy with an anytime approach, an intelligent path planner is developed which avoids static and pop-up obstacles effectively and provides a solution which is near optimal.

3.1. Standard RRT Algorithm

The objective of the path planner is to find a path from a starting position (x_{start}) to a goal position (x_{goal}) through a configuration space. Since the early development of mission planning algorithms, a large number of algorithms have been developed for such a problem. To name a few, one can find, road map, cell decomposition, potential field, visibility line [1],[9],[10]–[12]. However, solving the path planning problem while accounting for geometric and differential constraints is really challenging. LaValle [6] developed Rapidly-exploring Random Tree (RRT) as a potential solution for such problems. The standard RRT algorithm is given in Algorithm 1. This structure is the basis of the RRT algorithm, but there are several variations in the literature [6]–[7].

Algorithm 1 Standard RRT Algorithm

```
1: Choose an initial node $x_{init}$ and add to the tree $\tau$
2: Pick a random state $x_{rand}$ in the configuration space $C$
3: Using a metric $\rho$, determine the node $x_{near}$ in the tree that is nearest to $x_{rand}$
4: Apply a feasible control input $u$ to move the branch towards $x_{rand}$ at a pre-chosen incremental
   distance
5: If there is no collision along this branch, add this new node $x_{extend}$ to the tree $\tau$
6: Repeat steps 2 to 5 until $x_{goal}$ is included in the tree $\tau$
7: Find the complete path from $x_{init}$ to $x_{goal}$
```

3.2. Modified RRT Algorithm

The standard RRT algorithm provides a time-parameterized (open loop) set of controls to move x_{start} to x_{goal}. The accuracy of the resultant path depends on the validity of the state space model being used. In reality, apart from modelling uncertainties, there exist sensor inaccuracies, wind disturbances and other unmodeled factors. In [6], RRT is extended to generate paths in the output space. The modified RRT algorithm is as shown below.

The tracking of the generated waypoints depends on the feedback control policy. Because of inherited characteristics (incremental growth), the path generated by the above algorithm usually includes several extraneous waypoints, which is undesirable in the sense of the cost of travel along the path.

3.3. Incorporation of Actual Trajectory Characteristics

The true UAV trajectory deviates from the waypoint path [13], specially near the edges. These deviations can be predicted for the given kinematic model, and hence, collisions with obstacles can be determined. With this information, the path planning algorithm must ensure that the actual trajectory is also free from collisions. In [14], the RRTs are grown

Algorithm 2 Modified RRT Algorithm

```
1: Choose an initial node $w_{init}$ and add to the tree $\tau$
2: Pick a random waypoint $w_{rand}$ in the space $C$, with small probability, set $w_{rand} = w_{goal}$ to pull the
   graph towards the goal
3: Using a metric $\rho$, determine the node $w_{near}$ in the tree that is nearest $w_{rand}$
4: Extend the branch toward $w_{rand}$ by an incremental distance while taking care of the turn angle
   constraint
5: If there is no collision along this branch, add this new node $w_{extend}$ to the tree
6: Repeat steps 2 to 5 until $w_{goal}$ is included in the tree $\tau$
7: Find the complete path from $w_{init}$ to $w_{goal}$
```

assuming that the UAV can follow ± 45 degree change. This assumption leads to less oscillation near the edges but one has to assure that these oscillations do not collide with obstacles. This indirect way of incorporating the turn radius constraint into the algorithm affect the speed of algorithm severely.

To overcome this problem, we have developed a mechanism which enables the UAV to transit smoothly. If the UAV traverses to the waypoint k as shown in the Figure 1 before switching to waypoint $k + 1$, it will overshoot and may collide with obstacles, which is undesirable. It can be observed from the figure that these oscillations are significant for the sharp turn, which result in further danger to the mission. For the success of the mission, one has to predict the overshoots and check for collisions. The actual trajectory of the UAV can be parameterized in terms of straight lines and circular arcs. Since the next waypoint (randomly selected node) is known, we use this information to parameterize the actual trajectory of the UAV in the terms of straight lines and the arcs which satisfy the minimum turn radius constraint. The deviation due to the kinematic constraint now stays within the isosceles triangle $\Delta p1kp2$, and the path planner has to assure that there should be no obstacles in it. If there are obstacles, the randomly selected node is not incorporated in the tree. In this way, we have successfully converted differential constraint into geometric constraint with an additional obstacle for each new node. Moreover, this parameterizations gives a way to switch to the next waypoint smoothly.

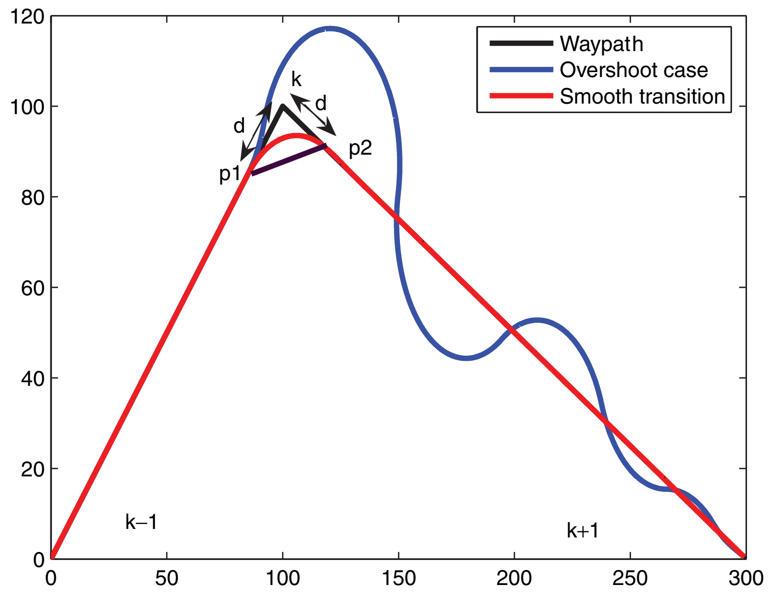

Figure 1. Smooth switching

The UAV starts taking the turn at the point p_1 and continues to the turn until it arrives at point p_2 as shown in Figure 1. With good path following technique, the UAV should be able to track the trajectory as long as the radius of arc is greater than the minimum turn radius R_{min}. We have developed a guidance law to track waypaths combining a pursuit guidance law with an LOS guidance law, which will be presented in the next section. The steps of the proposed approach incorporating actual trajectory characteristics are given in Algorithm 3.

The extraneous nodes are important in growing the tree can be eliminated because they do not provide any additional information. This process is carried out by smoothing the path (since we are interested in a path not in the whole tree), but this process does not reduce the path cost significantly. As already mentioned, because of random incremental growth of the RRT, the resultant path may not be straight even in the long corridors. To overcome this problem, a greedy strategy is proposed, exploiting the basic incremental growth characteristic while preserving random features. In this strategy, the incremental growth is continued until an obstacle is encountered, which is different from the modified RRT algorithm. This strategy will curb the tendency of growing too many path segments in the long corridors and will speed up the algorithm.

Algorithm 3 RRT Algorithm with incorporation of actual trajectory characteristics

```
 1: Choose an initial node $w_{init}$ and add to the tree $\tau$
 2: Pick a random waypoint $w_{rand}$ with little bias (set $w_{rand} = w_{goal}$ for the bias) in the search
    space $S$
 3: Using a metric $\rho$, determine the node $w_{near}$ in the tree that is nearest $w_{rand}$
 4: Extend the branch toward $w_{rand}$ by an incremental distance, resulting in node $w_{extend}$
 5: If there is no collision along this branch
 6: **if** check the collision with actual trajectory **then**
 7: add node $w_{extend}$ to the tree $\tau$
 8: **else**
 9: go to step 2
10: **end if**
11: Keep extending the branch by an incremental distance and keep adding new nodes $w_{new}$ to
    the tree $\tau$ until an obstacle is encountered
12: Repeat steps 2 to 11 until $w_{goal}$ is included in the tree $\tau$
13: Find the complete path from $w_{init}$ to $w_{goal}$
14: Eliminate extraneous nodes from this path while checking the turn radius constraint for
    actual trajectory
```

Although the greedy strategy curb the tendency of growing too many path segments, it can not be guaranteed that path cost will reduce significantly since the algorithm terminates when the target is first found. If enough computational time is available, more queries can be carried out to find a lower cost path; otherwise one has to accept the path found. The high cost of the path may outweigh the real time feature of the algorithm.

3.4. Towards a Suboptimal Solution

Since RRT is a general purpose technique, substantial effort is required to develop a suitable planner to satisfy mission requirements. In this section, we have attempted to develop an intelligent approach to generate suboptimal paths while preserving its real time applicability by combining actual trajectory characteristics with an anytime approach [8]. The anytime approach provides solutions for any given computational time and the solutions get better as the computation time increases. The idea is to generate a path using Algorithm 3 for a given scenario. When the UAV flies from one waypoint to another waypoint, the processor stays idle when the path is being tracked. The anytime algorithm exploits this idle position state and uses the available computation power to find a new less costly path. This strategy is followed until the UAV reaches the target point. The steps of the proposed intelligent approach are shown below in Algorithm 4.

Algorithm 4 RRT Algorithm for Suboptimal Solution

```
 1: Find a path for given space using Algorithm 3
 2: while Until UAV reaches to the target point do
 3:   Search for low cost paths using Algorithm 3 from next waypoint while tracking a line
      connected from waypoints
 4:   if An obstacle is encountered in waypoints path based on sensor range then
 5:     Search for a path from vicinity of UAV position
 6:   else
 7:     Keep tracking until UAV reached intended waypoint
 8:   end if
 9:   Select new waypoint based on low cost path
10: end while
```

It is clear that the proposed algorithm offers a useful approach for avoiding pop-up obstacles. The onboard sensor will keep checking whether obstacles exist on the flight path or not. If obstacles are detected, the UAV has to change its path to avoid a collision. Depending on the velocity of the UAV, the time to collide (T^c) is predicted and a new path must be found before T^c. Because of the quick exploration, the algorithm will find a path quickly, and hence, pop-up obstacles can be avoided. Note that the anytime approach will be implemented as soon as collisions are avoided. Moreover, the algorithm has the capability to deal with disturbances as well. If the UAV deviates significantly from its original path so that returning to the path may endanger the mission, a new search is carried out from the vicinity of the UAV flying area to find a new path.

4. UAV KINEMATICS AND GUIDANCE LAW

The UAV has to follow the path generated by the proposed approach. In order to track a given path, a guidance law is required which allows the UAV to track the path with minimum deviation. For this purpose, we employ a guidance law that is a combination of a pursuit guidance law and a line of sight (LOS) guidance law taken from the missile guidance literature [15]. This choice is motivated from the fact that pursuit allows the UAV to reach the desired waypoint quickly and it is simple to implement. However, the drawback is that its trajectory may not be along the LOS as desired. Hence, an LOS guidance law that will steer the UAV towards the LOS is needed to overcome this difficulty.

To design a guidance law, we need a navigation model of the UAV. For this purpose, we consider the following three state navigation model of a UAV

$$\dot{x} = V \cos \alpha \tag{1}$$

$$\dot{y} = V \sin \alpha \tag{2}$$

$$\alpha = \frac{a}{V} \tag{3}$$

where V represents velocity, α represents the heading angle and a represents lateral acceleration (latax) which acts as guidance parameter. The bound on latax is given as follow

$$-a_{\max} \le a \le a_{\max}$$

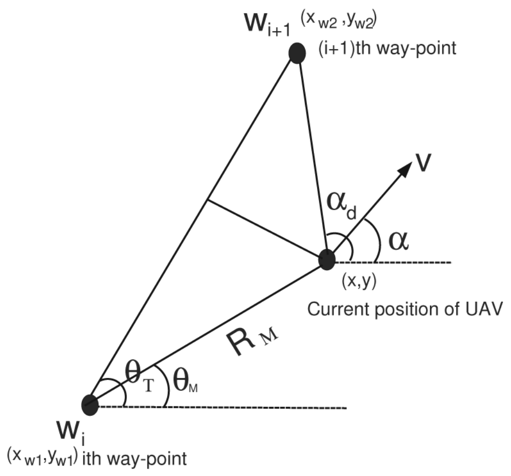

Figure 2. UAV guidance geometry

Consider the engagement geometry of the UAV as shown in Figure 2. Assume that the UAV was initially located at the x_{w1}, y_{w1} and it has to follow the path formed by the waypoints W_i and W_{i+1} from its current position x,y. For the UAV to move towards the waypoint W_{i+1}, it has to change its course so it aligns itself along the line connecting the waypoints. The required latax to align heading along LOS is calculated as follows

$$a = k_1 (\alpha_d - \alpha) + k_2 R_m \sin(\theta_T - \theta_M) \tag{4}$$

Here k_1 and k_2 are design parameters and other terms are defined as

$$\alpha_d \triangleq tan^{-1}\left(\frac{y_{w2} - y}{x_{w2} - x}\right)$$

$$\theta_T \triangleq tan^{-1}\left(\frac{y_{w2} - y_{w1}}{x_{w2} - x_{w1}}\right)$$

$$\theta_M \triangleq tan^{-1}\left(\frac{y - y_{w1}}{x - x_{w1}}\right)$$

where x_{w1}, y_{w1} and x_{w2}, y_{w2} are the x and y coordinates of the waypoints W_i and W_{i+1} respectively. The first term in (4) represents a pursuit policy which guides the UAV towards the way-point whereas the second term in (4) represents the LOS policy which tries to minimize separation from the LOS.

As mentioned in Section 3.3, in order to traverse smoothly to the next waypath, the UAV has to start turning before it arrives to the next waypoint. Consider the waypath geometry as shown in Figure 1. Assume that the UAV is approaching towards waypoint k and it has to move to waypoint k + 1. It has to start to turn at point p_1 and follow the arc until it reaches point p_2. The distance d of points p_1 and p_2 from waypoint k is a function of arc radius and the angle between the waypath as follows

$$d = \frac{R}{\tan\beta} \tag{5}$$

where 2β is the angle between waypath formed by joining waypoint k – 1 to k and k to k + 1. The guidance law to follow the arc is different from to follow a straight line, since LOS term is not required to follow the arc. The distance d in equation (5) provides the measure for switching from pursuit law to combined pursuit and LOS laws and vice versa. This switching will enable the UAV to traverse without any overshoots.

Performance of the Guidance Law

A guidance law is required to follow the waypath generated by the path planning algorithm. Towards this direction, we have developed a guidance law with a switching mechanism which enables the UAV to follow the waypath smoothly. The switching mechanism uses a measure which is a function of turn radius and angle between consecutive waypaths. Figure 3 depicts the performance of the guidance law in which waypaths are right angle to each other. It can be observed from the figure that UAV starts turning before it reaches the waypoint to traverse the next waypath smoothly. After it completes the turn, it follows the desired straight line with minimum deviations.

5. SIMULATION RESULTS

In this section, we present simulation studies to demonstrate the performance of the proposed algorithm. The objective is that a UAV has to travel from a given starting location to a goal location while avoiding static and pop-up obstacles in an obstacle rich environment. It is assumed that the UAV is flying at constant altitude.

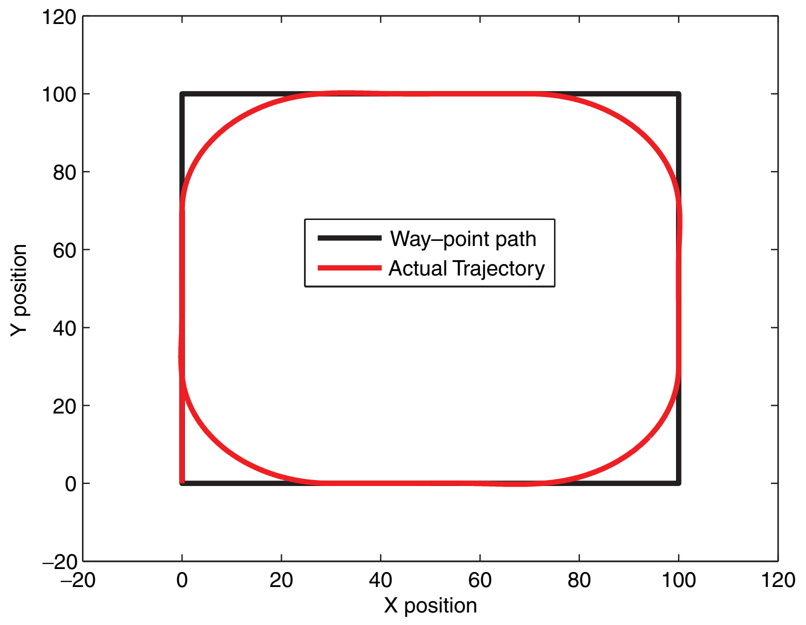

Figure 3. Performance of the guidance law

As already argued, when the turn radius of the UAV is comparable with a corridor of the obstacles, through which the UAV has to travel, it is important to consider the turn radius constraint. Otherwise, the UAV may collide with obstacles in the corridor. RRT has the capability to handle kinodynamics or nonholnomic constraints while adding a node in the tree. We have taken advantage of this feature and made sure that the UAV transits smoothly from one waypath to another waypath. In order to this, we have considered the guidance characteristic in the path planning algorithm and made sure that actual trajectory of the UAV does not collide with obstacles. As mentioned in Section III, branches of the tree need to be extended in randomly chosen directions. The unit by which each branch of the tree increments must assure that the initiated turn is completed before the new turn starts. With this in mind, the minimum incremental distance is chosen as three times the turn radius distance.

Now we will demonstrate the effectiveness of the path planning algorithm based on the following example scenarios. The first example will show the ability of the proposed algorithm to generate a path, taking the constraints into account in an obstacle rich environment (Section 5.1). The second example will demonstrate the capability of the algorithm to deal with pop-up obstacles

(Section 5.2). In third example, it shows how solutions improves as the mission progresses (Section 5.3). Finally, the last example presents the computational requirement analysis for various scenarios (Section 5.4).

5.1. Path Planning in Obstacle Rich Environment

As discussed before, the algorithm has the capability to provide collision free paths while satisfying a turn radius constraint. We have tested the performance of the path planner with several scenarios. The position and size of obstacles are chosen randomly, however, lengths and widths of obstacles are kept above the prefixed minimum length and width. Each time the path planner finds the flyable path quickly. In Figure 4, we find a path in a terrain of 5 × 5 km^2 with 1500 obstacles. It can be observed from the figure that the path is free from collision. The tracking performance of the path is checked by deploying the guidance law developed in Section 4. Planned and executed trajectories are almost on top of each other. The closeness of the trajectories verify the benefits of incorporating actual trajectory characteristics into the path planner algorithm.

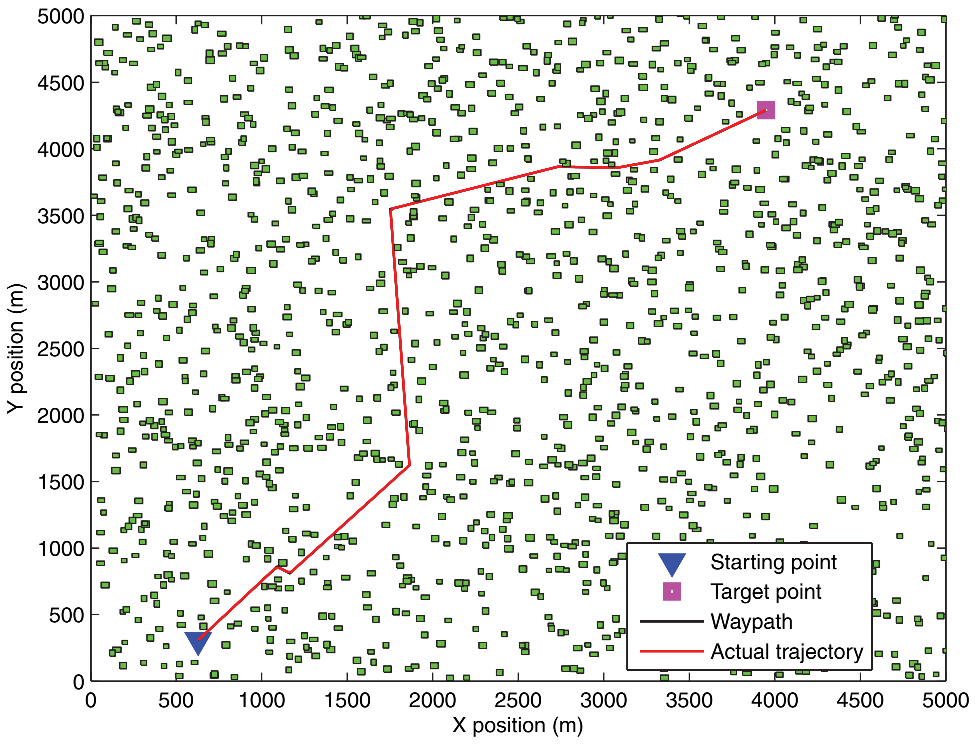

Figure 4. Performance of path planner in obstacle rich environment

5.2. Path Planning with Anytime Approach

There are serval considerations for an ideal path planner including optimality, completeness and computational complexity. In general the optimal solution demands high computation, and hence cannot be computed in real time. The RRT algorithm generates a quick solution but it is far from optimal due to its random exploration of the space. The idea here is to generate suboptimal solutions while preserving quickness of the algorithm. This is achieved with the anytime approach to develop a suboptimal path planner which not only provides a solution quickly but the quality of solution gets better as the computation time increases. The steps of the proposed algorithm are given in Algorithm 4. The anytime approach uses the idle state of the processor, when the UAV flies between two given waypoints, to find new paths. The search is carried out during the time it takes to fly from one waypoint to the next. If the cost of any new path is less than the previous path it is accepted; otherwise the UAV follows the earlier path. This search is continued until the UAV reaches the target.

5.3. Path Planning With Pop-up Obstacles

Obstacles are classified into two types: a priori known obstacles (e.g. from a terrain map) from which a trajectory can be generated before flight, and pop-up obstacles encountered during flight. A terrain map is usually used to generate initial waypoints, and a local path planning scheme such as bouncing [9] is used to avoid pop-up obstacles. Here we deal with both types of obstacles through the proposed algorithm because it is able to provide solutions very quickly. It is assumed that the UAV is mounted with onboard sensors which can detect obstacles if they are on collision course. As soon as the UAV detects an obstacle, the algorithm incorporates this information into the terrain map and starts searching for a new path. Figure 5 shows such a scenario in which pop-up obstacles are encountered twice in the waypath. The algorithm finds a new path in 0.04 sec and 0.06 sec respectively. The pop-up obstacles are shown in red blocks in the figure whereas green blocks represent static obstacles. The blue lines show the new path after pop-up and the red line represents the actual trajectory of the UAV. The third pop-up obstacle does not appear on the path and therefore there is no need to search for a different path. As soon as a new waypath is generated, it is tracked effectively by the UAV using the guidance law given in Section 4.

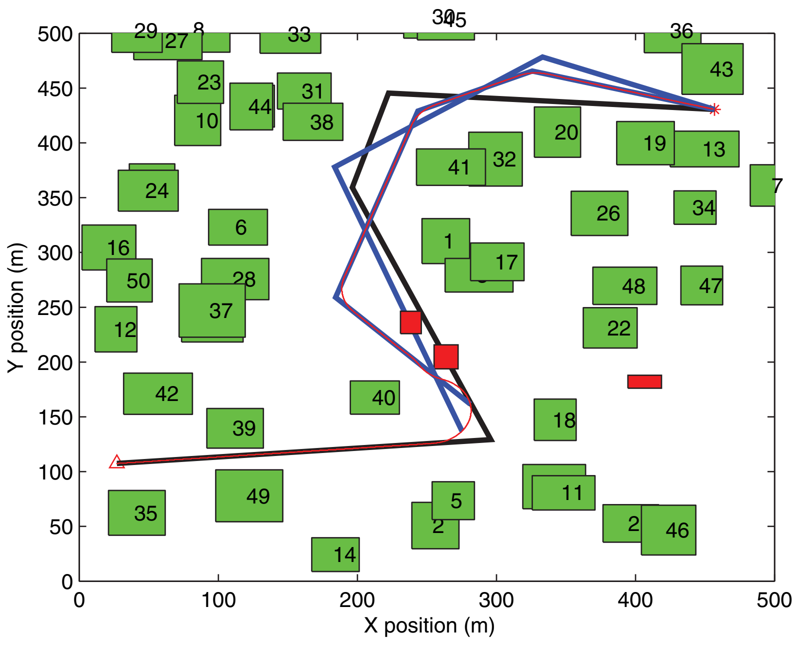

Figure 5. Path planning with pop-up obstacles

Figure 6 shows the trajectories with and without the anytime approach strategy. It can be observed that during the flight the proposed algorithm always tries to find a low cost path to the next waypoint. Note that although we have considered a one waypoint ahead search strategy, it is not always necessary that the search is carried out up to the next waypoint. One can choose intermediate points as well and can find new paths for use once the intermediate path has been reached.

5.4. Computational Performance of the Path Planning Algorithm

The proposed path planning algorithm can provide a solution in real time as shown in this study The simulations are carried out using MATLAB 7.0 on a Pentium 4 (2.4 GHz) machine with 2 GB RAM. In the simulations, we have kept the starting and the target positions the same in an environment of $5\ km \times 5\ km$ with 500,750 and 1000 obstacles respectively. The computation time to find a path is recorded and the results are summarized in Table 1. It can be observed that even in the worst case the computational time required is of the order of 10 sec. Moreover, the computation time requirement does not increase significantly as the number of obstacles increases. Hence the proposed scheme can be useful for real time applications.

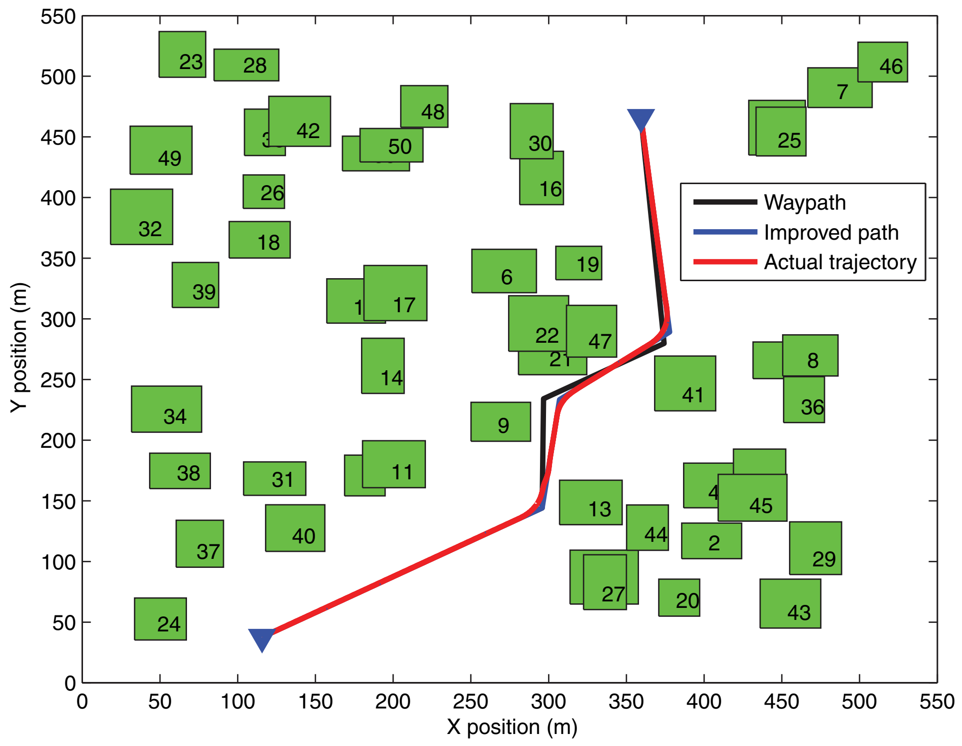

Figure 6. Towards suboptimal solution

Table 1. Computational time requirement for different scenarios

		Computational time (sec)	
Number of obstacles	Minimum	Maximum	Averaged over 20 simulations
500	0.50	2.12	1.21
750	0.82	2.98	1.44
1000	1.53	5.65	3.11

6. CONCLUSIONS

In this paper, a suboptimal path planing strategy for a UAV is proposed for obstacle rich environments. The proposed method incorporates actual trajectory characteristics with anytime approaches. The algorithm performs well and provides collision free paths in the presence of static and popup obstacles. The algorithm finds a path in a reasonably short time and the quality of solution improves as computational time increases. The performance of path planner is verified by various simulation studies and is found to be satisfactory.

7. ACKNOWLEDGEMENT

The work is carried out under the UKIERI Programme.

8. REFERENCES

[1] J. Latombe, "Robot Motion Planning," *Boston: Kluwer Academic Publishers*, 1991.

[2] S. M. LaValle, "Planning Algorithms", *Cambridge University Press*, 2006.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, "Principles of Robot Motion: Theory, Algorithms, and Implementations", *Cambridge, MA: MIT Press*, 2005.

[4] N. M. Amato and Y. Wu, "A Randomized Roadmap Method for Path and Manipulation Planning," *in Proceedings of the IEEE International Conference on Robotics and Automation*, 1996, pp. 113-120.

[5] L. E. Kavraki, P. Svestka, J. C. Latombe, J. C. and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces". *IEEE Transactions on Robotics and Automation*, Vol. 12, 1996, pp. 566-580.

[6] S. M. LaValle, "Rapidly-exploring Random Trees: A New Tool for Path Planning," 1998, TR 98-11, *Computer Science Dept., Iowa State University*.

[7] J. B. Saunders, B. Call, A. Curtis, R. W. Beard, and T. W. McLain, "Static and Dynamic Obstacle Avoidance in Miniature Air Vehicles", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.

[8] D. Ferguson and A. Stentz, "Anytime, Dynamic Planning in High-dimensional Search Space", *in Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1310-1315.

[9] D. W. Gu, I. Postlethwaite and Y. Kim, "A Comprehensive Study on Flight Path Selection Algorithms", *Dept. Of Engineering, University of Leicester*.

[10] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *in Numerische Mathematik. Springer*, Vol. 1, 1959.

[11] 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, 1968.

[12] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-query Path Planning," *in Proceedings of the IEEE International Conference on Robotics and Automation*, 2000, pp. 995-1001.

[13] E. Frazzoli, M. A. Dahleh and E. Feron, "Real-time Motion Planning for Agile Autonomous Vehicles," *AIAA Journal of Guidance, Control and Dynamics*, Vol. 25, No. 1, 2002, pp. 116-129.

[14] M. Kothari, D.-W. Gu, and I. Postlethwaite, "An Intelligent Sub-optimal Path Planning Algorithm Using Rapidly-exploring Random Trees", accepted to European Control Conference 2009

[15] P. Zarchan, Tactical and Strategic Missile Guidance, *AIAA Inc.,* Washington, 1990.