3-D Path Planning for the Navigation of Unmanned Aerial Vehicles by Using Evolutionary Algorithms

Isil Hasircioglu
Computer Engineering Dept.
Marmara University
Goztepe, Istanbul, Turkey
hasircioglu@eng.marmara.edu.tr

Haluk Rahmi Topcuoglu
Computer Engineering Dept.
Marmara University
Goztepe, Istanbul, Turkey
haluk@eng.marmara.edu.tr

Murat Ermis
Industrial Engineering Dept.
Turkish Air Force Academy
Yesilyurt, Istanbul, Turkey
ermis@hho.edu.tr

ABSTRACT

Military missions are turning to more complicated and advanced automation technology for maximum endurance and efficiency as well as the minimum vital risks. The path planners which generate collision-free and optimized paths are needed to give autonomous operation capability to the Unmanned Aerial Vehicles (UAVs). This paper presents an off-line path planner for UAVs. The path planner is based on Evolutionary Algorithms (EA), in order to calculate a curved path line with desired attributes in a 3-D terrain. The flight path is represented by parameterized B-Spline curves by considering four objectives: the shortest path to the destination, the feasible path without terrain collision, the path with the desired minimum and maximum distance to the terrain, and the path which provides UAV to maneuver with an angle greater than the minimum radius of curvature. The generated path is represented with the coordinates of its control points being the genes of the chromosome of the EA. The proposed method was tested in several 3-D terrains, which are generated with various terrain generator methods that differ with respect to levels of smoothness of the terrain.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*Heuristic Methods*; J.7 [**Computers in Other Systems**]: Military

General Terms

Algorithms, Experimentation

Keywords

Unmanned Aerial Vehicles, Path Planning, Genetic Algorithms, B-Spline Curves

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have been recently used in commercial applications [1] as well as in military areas for different missions including weather reconnaissance, search and rescue assisting operations in sea and mountains, aerial photographing and mapping, fire detection and traffic control. The most important reasons of the popularity of the UAVs are low force need, no vital risk by performing the most hazardous missions and long endurance.

Autonomous operation of UAVs requires the development of control systems that can work without human support for long time periods. The vehicles are required to make both low-level control decisions, such as path planning, and high-level decisions, such as cooperative task assignment. Task assignment is crucial for designing successful missions in heavily defended environments while the path planners which generate collision-free and optimized paths are needed to give autonomous operation capability to the UAVs. Both aspects of the fleet problem of the UAVs build up the optimal flight while by deciding resource allocation and determining the best trajectory.

There are many different approaches to path planning including potential fields [2], A* graph search algorithm [3], differential evolution [4], linear programming [5]. Genetic Algorithms have been used to solve path planning problem of ground vehicles. While there are works in which the binary representation and classical GA operators have been used [6], floating point representation has been used in [7] by using more sophisticated GA operators. Moreover, improved genetic algorithms which tries to overcome the problems of the traditional GA have been applied on the path planning of the mobile robots [8].

Evolutionary Algorithm gives successful results for 3-D UAV path planning problem. Simple spline path representation with different path mutation mechanisms is simulated in [9]. On the other hand, B-Spline curve representation is applied to represent the flight path of the UAVs in [10, 14], where they generate terrain using only mathematical functions which requires a large set of experimentally predefined constants.

In our work, an offline path planner of unmanned aerial vehicle navigation is proposed for 3-D terrain structures that are constructed by various terrain generator algorithms. The flight path is represented by parameterized B-Spline curves. We update the safety distance calculation given in [14] by considering the semi-sphere region below the UAV and se-

lecting the minimum one. Another difference of our work is to present mission specific goal definitions such as sweeping flights, flight envelope in a given line of sight etc. We consider a more realistic model for the maneuver capability and technical specifications of UAVs. Specifically, maximum altitude constraint is considered as part of the fitness function evaluation. We also present problem specific mutation operators which consider the feasibility of the path in order to include new control points. In our study, the number of control points can also be updated adaptively in order to analyze the terrain more accurately.

The rest of the paper is organized as follows. The next section describes the problem formulation including terrain generation, path representation, objectives and constraints. In Section 3, we present the implementation details of the Genetic Algorithm approach for the path planning problem. Section 4 summarizes results of the experimental study with respect to selected terrains. Finally, conclusion and future work are presented in Section 5.

2. PROBLEM DEFINITION

In this paper, we consider the path planning problem of the Unmanned Aerial Vehicles in a 3-D environment. To simulate a 3-D environment, jME (jMonkey Engine) [11], a Java scenegraph API based on OpenGL, is used. The API is easily adapted to generate and visualize terrain since it's open source and has many features. Landscape generation phase creates heightmap data by different methods in different sizes. A heightmap is an image used in 3-D computer graphics to store values. Heightmap is generally represented by a two-dimensional matrix of which dimensions represent coordinate axis, mostly X and Z. The value of the matrix's entry gives the height value, Y coordinate, of the corresponding point. The scale of the matrix which gives the point intervals may be changed by setting appropriate values. The height value of a point may be found via this formula:

$$height = heighmap[(column + row * size)] \quad (1)$$

where *column* and *row* are dimensions of the heightmap matrix and *size* is the size of the matrix. The height values of the points which are not stored in the heightmap matrix due to the scale may be found with linear interpolation via the formula:

$$height = height_1 + \frac{desired\_point - point_1}{point_2 - point_1} * (height_2 - height_1) \quad (2)$$

where *desired_point* is the point of which height value to be calculated but not stored in the heightmap, $point_1$ and $point_2$ are the closest points to the desired point stored in the heightmap matrix, $height_1$ and $height_2$ are the height values of the $point_1$ and $point_2$ respectively.

2.1 Path Representation

To apply genetic algorithms to the path planning problem, the path needs to be encoded into genes to represent an individual in the population. For better representation of the path line, floating point coding is used in our work. Since it's not efficient way to represent the path line of a UAV by straight line segments as in mobile robot applications [6] or simple splines as in [9], the path is represented by using B-Spline curves.

B-Spline curves are parametric curves constructed by calculating basis functions [16]. Mathematically, given $n+1$ control points P_0, P_1, ..., P_n and a knot vector $U = u_0, u_1, ..., u_m$, the B-Spline curve of degree k defined by these control points and knot vector is

$$C(u) = \sum_{i=0}^{n} N_{i,k}(u) * P_i \quad (3)$$

where $N_{i,k}(u)$'s are B-Spline basis functions of degree k. With coordinates $(x_0, y_0, z_0), \ldots, (x_n, y_n, z_n)$, the coordinates of the B-Spline can be written as:

$$X(t) = \sum_{i=0}^{n} N_{i,k}(u) * x_i \quad (4)$$

$$Y(t) = \sum_{i=0}^{n} N_{i,k}(u) * y_i \quad (5)$$

$$Z(t) = \sum_{i=0}^{n} N_{i,k}(u) * z_i \quad (6)$$

The degree of the B-Spline curve determines the smoothness of the curve. When the value of k becomes higher, it will cause smoother curves. In this study, the degree is taken as 3 in order to provide required smoothness.

The B-Spline basis functions are used as weights like in Bezier curves. The function domain is divided by knots and the basis function values have always non-zero values. The basis functions $N_{i,k}$ are defined recursively with an algorithm (called as de Boor recursion formula) using the knot values. The details of B-Spline curve constructions including the recursion formula can be found in [16].

In our path representation, the coordinates of the control points which have floating point values, form the individuals' chromosomes. The first (source) and the last (destination) points of the curve are fixed. The dynamic control points between these points which forms the route of the UAV are used to construct the structure of the individual. B-Spline curves are suitable to represent the path of the UAVs since they need few variables to define complicated curve paths. Moreover, an update in one of control points changes only the area near the updated control point due to its local effect.

2.2 Objectives

The problem of computing the optimum navigation path of an UAV is formulated as a minimization one, which takes into account five general constraints that are listed below:

1. Constructing feasible paths without terrain collision,
2. Constructing a path within the desired minimum and maximum distance to the terrain,
3. Constructing a path which provides UAV to to maneuver with an angle greater than minimum radius of curvature,
4. Minimizing the length of the path,
5. Targeting mission specific objectives (providing sweeping flights or providing a flight envelope in a given line of sight).

These constraints can be classified into two categories: *hard constraints* (or *technological constraints*) that must be satisfied at the generated flight path, and *soft constraints* that are evaluated in terms of the given measure for the quality of the solution. The first three constraints listed above are hard constraints and the remaining ones are the soft constraints. In this paper, the first four constraints are considered for path planning.

The quality of the constructed path by considering the above constraints of an UAV is measured by a fitness function value calculated as the sum of four different cost terms:

$$f = \frac{1}{\sum_{i=1}^{4} w_i * f_i} \tag{7}$$

where f_is are cost function values described below; and w_is are weights of each function determining the priority of the function. Note that each f_i is associated with the corresponding constraint given above.

The term f_1 in Equation 7 represents the number of curve points inside the solid boundary. So, the penalty value is proportional to the number of discretized curve points located under the solid surface. All n path segments of the flight path are checked whether or not pass through the terrain by computing the distance between the curve point and the terrain using heightmap data. If this is true for a discrete point of the path line, a constant penalty is added to term f_1. Consequently, this term provides to construct the collision-free paths by giving high penalty values to the curves with more points inside the solid boundary. The formulation of the cost f_1 is as follows:

$$f_1 = \sum_{i=1}^{n} y_i, \quad where \quad y_i = \begin{cases} 1 & \text{if } d_i <= 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where d_i is the distance between the curve point and the terrain, and i is the index of curve points.

The term f_2 is relevant to the distance penalty of the curve. This term actually consists of two penalty values including safety distance penalty (keep a safety distance between UAV and the ground) and maximum distance penalty (maximum altitude because of technological restrictions or mission specific constraints). Safety distance penalty aims to provide a flight path far from specified minimum distance. The distance for each discretized curve point is checked for the safety distance and penalty value proportional to the difference between the distance and desired safety distance is calculated. The penalty value for each curve point forms the cost function value. However, to calculate the distance of the curve point for only one terrain point cannot be sufficient in 3-D environment since the curve point can be closer to the any other points than the corresponding terrain point. Therefore, for each curve point, the distance between the corresponding terrain point and its neighbor points is calculated; the maximum penalty value of the neighboring terrain point distances is considered for each curve point as the safety distance penalty value.

Maximum distance penalty provides a flight path that is in the range of specified maximum altitude of the UAV. If this term is ignored, the generated path might be the shortest one but that path can not be realized with respect to UAV's technical specifications. The penalty value is calculated proportional to the difference between the distance and allowed maximum altitude; i.e.,the far the curve point is the larger penalty value it has. The term f_2 given below is represented in two separate terms, $f_{2,1}$ and $f_{2,2}$, which are formulated below:

$$f_{2,1} = \sum_{i=1}^{n} \max_{k \epsilon N(.)} \left\{(d_{min} - d_{i,k})/d_{min}\right\} * y_i,$$
$$where \quad y_i = \begin{cases} 1 & \text{if } d_{i,k} <= d_{min} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$f_{2,2} = \sum_{i=1}^{n} \left\{(d_i - d_{max})/d_{max}\right\} * y_i,$$
$$where \quad y_i = \begin{cases} 1 & \text{if } d_i >= d_{max} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$f_2 = f_{2,1} + f_{2,2} \tag{11}$$

where $N(.)$ is the neighbor set of the specific terrain point, $d_{i,k}$ is the distance between the curve point and the terrain point, d_{min} and d_{max} are the minimum and maximum safety distance from the terrain respectively.

The term f_3 indicates the minimum radius of curvature penalty of the path. It is designed to prevent the UAV from exceeding the lateral and vertical acceleration limits, since the flight envelope determines the maximum radius of turns for the UAV. Its aimed to achieve a flight path with a specified minimum curvature angle by this term. To obtain penalty value, angle between two line segments of the curve is calculated for each three neighbor curve points. As in previous terms, penalty value is added proportional to the difference between the angle and specified minimum angle. The term f_3 can be formulated as the following:

$$f_3 = \sum_{i=1}^{n} \left\{(\Theta - \theta_i)/\Theta\right\} * y_i, \quad where \quad y_i = \begin{cases} 1 & \text{if } \theta_i <= \Theta \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where θ_i is the angle between the control point and the neighbors, and Θ is the minimum curvature angle.

The term f_4 is the length of the flight path calculated by adding the distance of two neighbor curve points for each discretized point. This term is used in order to minimize the flight path lengths and can be formulated as:

$$f_4 = \sum_{i=0}^{n} \left[(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2 \right]^{1/2} \tag{13}$$

where n is the curve discretization number; x, y, and z values are discrete coordinates of the points on the curve.

In order to evaluate fitness value, all f_is are normalized by using Equation 14 to get values in the same range after their values are calculated. This normalization transforms f_i values into the range of $[0, 1]$.

$$normalized\_value = \frac{value - value^{min}}{value^{max} - value^{min}} \tag{14}$$

where $value$ is the real value of the function, $value^{min}$ is the minimum value which the function can take and $value^{max}$ is similarly the maximum value which the function can take. Since the objective is to minimize the cost function values, the fitness function f in (7) is the inverse of the weighted sum of the normalized cost functions.

The weights (w_is) given in the main equation can be experimentally determined or can be set according to different

aspects such as terrain specifications, mission specific goal definitions (i.e., sweeping flights, flight envelope in a given line of sight) etc. As the main objective is to obtain feasible flight paths, weights may be determined in such a way that the corresponding term (one of the f_is) dominates the rest. In our experimental study, equal weights are considered.

3. EA-BASED SOLUTION FOR PATH PLANNING PROBLEM

In this section, we briefly explain the details of our evolutionary algorithm which generates a 3D path according to the provided constraints in a given terrain.

3.1 Representation of Individuals

In our study, each individual which is a path of the UAV is represented by B-Spline curve control points. B-Spline curve is a parametric curve which is described by control points. Each chromosome has control points which have x,y and z coordinates with floating point values in 3-D environment case. The first and the last points of the individual represents source and destination points, respectively. These two points are fixed for all individuals; the other free-to-move control points, which are located in ordered manner in the chromosome, may take different values. The number of control points may differ in different populations or in different individuals in a same population.

3.2 Initial Population

Initial population of the evolutionary algorithm in our problem is generated randomly by considering the boundaries of the terrain. The x and z coordinates are selected randomly from the range $[0..TerrainSize-1]$; and the y coordinate of the point, which will draw the height of the individual, is a set randomly by considering the feasibility in that case. To be able to help the feasibility of the curve, the y coordinate of the control points are generated in upper side of the terrain. If a non-feasible point which collides the terrain is created, its y coordinate is re-generated by random number generator until it provides feasibility of the point. This does not guarantee the collision-free B-Spline curve since its curve points may collide the terrain although its control points are feasible; but, this method increases the probability of feasibility of the curves.

3.3 Crossover and Mutation Operators

One-point crossover is considered in our study. This operator simply cuts the parents from one point randomly, and splits both parents at this point; then it creates two new individuals by exchanging the portions left.

New knowledge based mutation operators which work on the free to move control points of the individual are proposed. The operators alter the number or the position of the control points according to the fitness of the control point under consideration. The B-spline curve is divided into segments, which are the separated parts between the control points. In general, the operator starts working by finding the worst segment namely the segment which has the minimum fitness (maximum cost) of the individual curve. While calculating the fitness of the curve segments, the length term in the objective function is not considered, since it does not effect the feasibility of the curve. We consider three different mutation operators in our experiments.

- *Update.* This mutation operator updates the worst control point of the individual. The quality of each control point is computed by the aggregate cost of the curve points on the two neighbor curve segments. For instance, the cost of the control point 2 is calculated by adding the cost of the segment 1 and segment 2, where segment 1 is the piece of the B-Spline curve which is between the control points one and two, and the segment 2 is the piece of the B-Spline curve which is between the control points two and three. After the control point which has the maximum cost value is found, and if it is not the source or the destination point, this control point is replaced by a new one in order to increase the fitness of the curve. In order to achieve this objective, the new point is created in a feasible region. The x and z coordinates of the new point are set by considering the following equations

$$x = x + k * \Delta x + \epsilon \quad (15)$$

$$z = z + k * \Delta z + \epsilon \quad (16)$$

  where k and ϵ are terrain-specific parameters and Δx and Δz are the difference between the neighbor control points for x and z axis, respectively. The y coordinate of the new control point is randomly set from the feasible region generated by using the new x and z values. The y coordinate is set by adding a displacement to the current value of y so that final value will be within the minimum and maximum safety distances.

- *Insertion.* This mutation operator increases the number of the control points in a given individual and targets to increase the quality of the individual. After getting the fitness values of each curve segment, the segment which has the lowest value is established. Although update operator considers aggregate values of segments in order to update a control point, the insertion operator considers individual segments in order to insert a new control point. In this operator, first the worst curve segment is determined, then the curve point on this segment which has the maximum cost is found. By using the x and z coordinates of the selected curve point, a new control point of the B-Spline curve is created dynamically in a feasible region by considering the safety distance constraints.

- *Deletion.* This operator also alters the number of the control points in the individual. It finds the worst control point in the B-Spline curve (by considering aggregate fitness values of segments as in the update operator) and simply deletes this point from the curve. Therefore, this individual has one less control point number. Figure 1 shows how to apply the three mutation operators on a given B-Spline curve.

3.4 Parent and Survivor Selection

The selection of individuals for mating is carried out by tournament selection method [15]. This selection process has two stages:

- Select a number of k individuals randomly

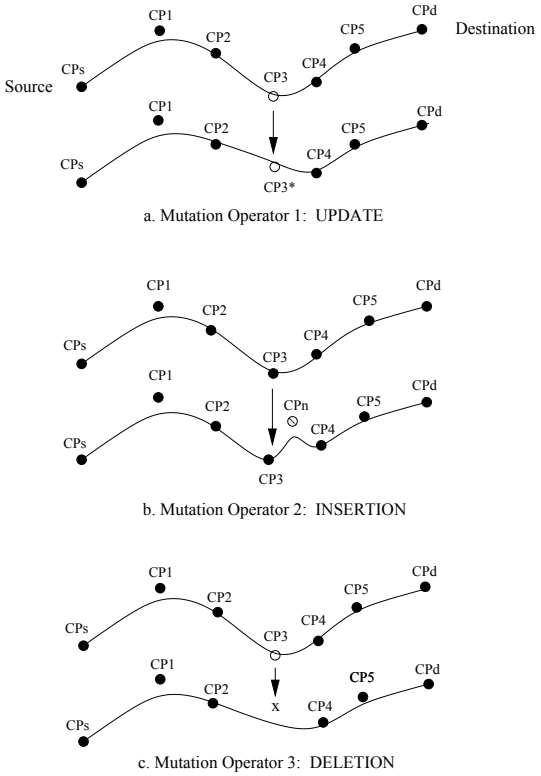

Figure 1: The Types of Mutation Operators.

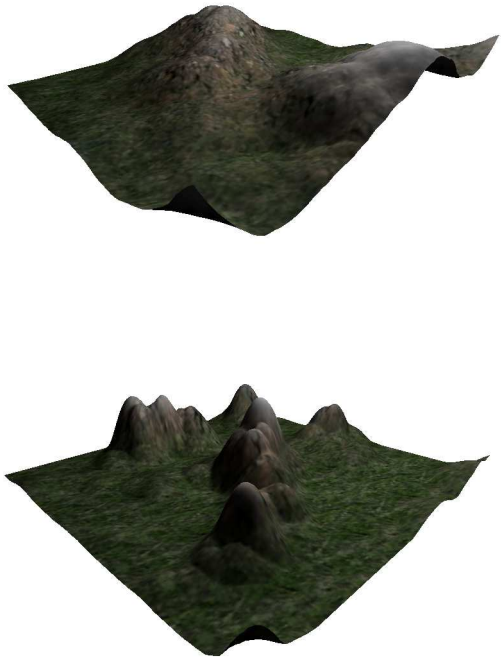

Figure 2: Sample Terrains: Terrain 1 (generated by the Midpoint Displacement Algorithm) and Terrain 2 (generated by the Hill Algorithm)

- Select the parent with the best fitness among selected k tournaments

where k is the tournament size, which is a variable parameter of the Genetic Algorithm phase. In our implementation, different tournament sizes are considered during the pre-experimentation phase; and k is set to 5 according to the results of experiments.

In our study, a steady-state population model is considered with an elitism-based survivor selection mechanism. At each iteration or generation, crossover and mutation operators are applied on selected individuals in order to generate two offsprings. Then, the best offspring among them is selected and it is put in place of the worst individual in the population.

4. EXPERIMENTAL STUDY

In our experiments, two terrain generator methods are selected as test environments including the Midpoint Displacement algorithm [12] and the Hill algorithm [13]. To test various terrain types, these two algorithms are used by changing algorithm parameters. Mainly, 128x128 terrain size is considered by using a set of additional arguments in order to set roughness or smoothness of the terrain. Two terrains given in Figure 2 are among the sample terrains considered in our comparison study, where first terrain is generated by using the Midpoint Displacement algorithm and the second one is generated by using the Hill algorithm.

An experimental study is done in order to identify the settings of various GA-parameters. Based on the experiments, population size is set to 200, the tournament size is set to 5; and the crossover rate is set to 0.8. In our experiments, we consider equal values for function weights given in Equation 7. The threshold values for distance and curvature radius are set according to the terrain height values. It is observed that different mutation rates give better results for different mutation types. Specifically, update mutation gives its best performance when the mutation rate is equal to 0.10; and, the random insertion/deletion mutation gives its best performance when mutation rate is equal to 0.15.

In our first set of experiments, the effect of generation size on the performance is observed by getting the best and the average fitness values. We set the GA parameters with the observed values given in the pre-experimentation phase, unless otherwise specified. We consider update mutation operator for the first set of experiments. Additionally, the number of control points are set with the values observed in the pre-experimentation phase. Specifically, a chromosome with 5 control points (including the source and destination points) gives best results for the first terrain (Terrain 1); and solutions with 7 control points outperforms other alternatives for Terrain 2. In order to measure the performance of varying the numbers of control points, minimum number of control points is set to 5. The effect of the number of control points on the quality of the solution is also measured as part of our experimental study.

It was observed that our algorithm provides better performance with the increase in the number of generations. After a specific point, the performance can not increase as previous increments while the running time increases almost linearly. While average fitness of the population continues

increasing as shown in Figure 3, the best fitness value becomes constant after a point (see Figure 4). We consider these limit values (different values for different terrains) as generation size for the later experiments.

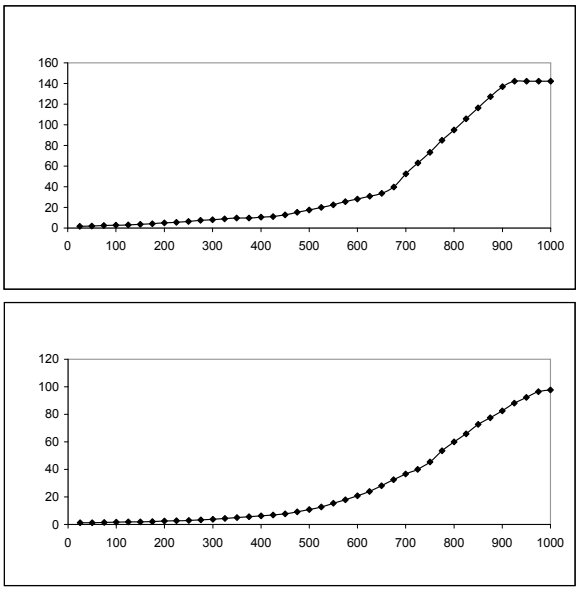

Figure 3: Average fitness value of each generation for two terrains

We measure the feasibility of the results for the two different terrains as well. Individual feasibilities of the best solution (with respect to three constraints of the given objective) are measured at each generation (see Figure 5). It's observed that collision penalty and curvature radius penalty values are all zero beginning from the first generation; i.e. it constructs feasible paths without terrain collusion where UAV maneuvers with an angle greater than minimum radius of curvature. On the other hand, safety distance penalty (which is set when the constructed path is not within the desired minimum and maximum distances to the terrain) decreases to zero value at the end of the [800th-1000th] generation range.

As generation size grows, not only the feasibility of the best individual increases but also number of feasible individuals in a given population increases. As shown in figure 6, while whole population of 200 individuals become feasible with respect to curvature radius and collision at the end of the GA run, most of the individuals still remain infeasible with respect to the safety distance.

The number of control points in the B-spline structure is also varied in order to measure its effect on solution quality. For easier terrains (i.e. normally one or two wide hills), smaller number of control points have higher performance; and fitness value of the best individual in a population generally decreases as number of control points increases. In an experiment, the number of control points is set from the range [5..11]. For the 128x128 size terrains, it seems that the number of control points selected for test phase are enough to observe the effect of the different number of control points. It's obviously seen that the most successful individuals have 5 control points in case of easier terrains as shown in fig-

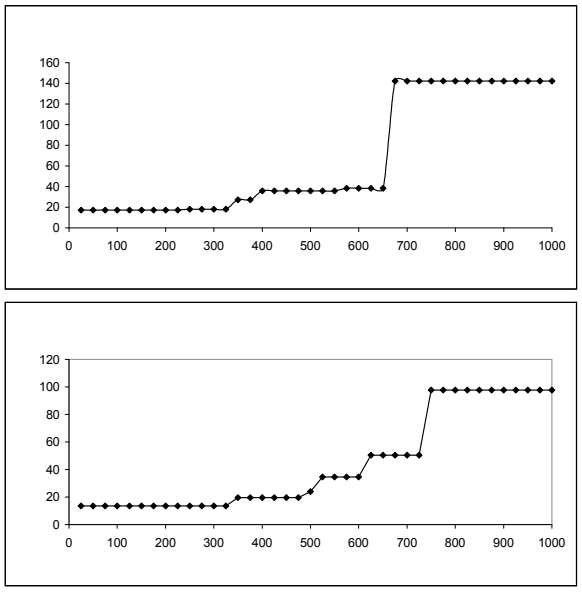

Figure 4: Best fitness value of each generation for two terrains

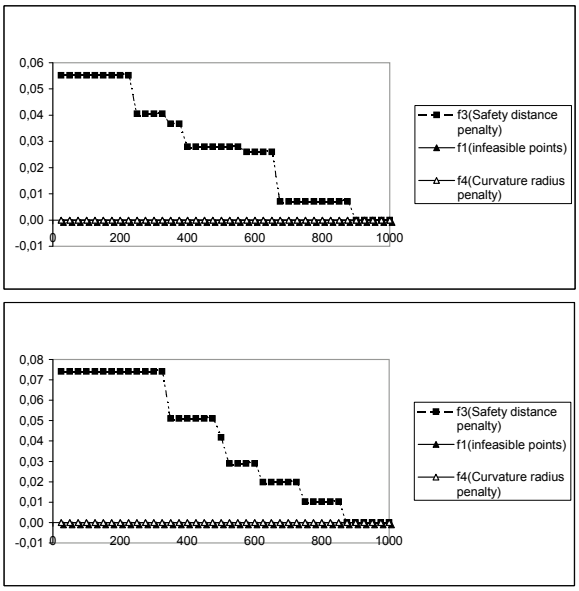

Figure 5: Feasibility of the best solution in each generation by considering three constraints of the unified objective

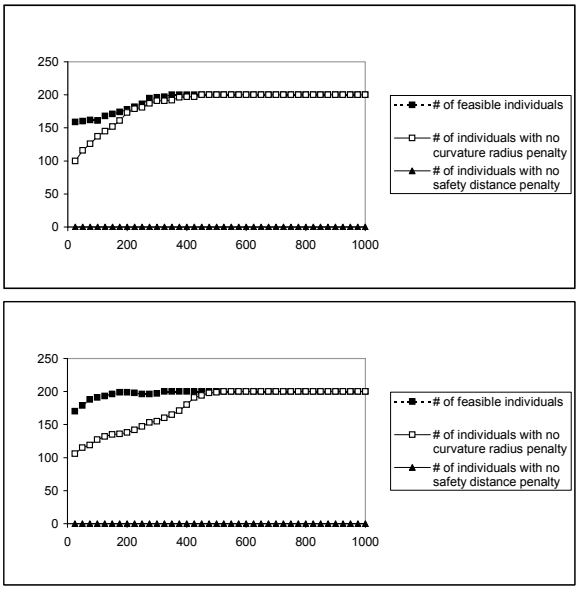

Figure 6: Number of feasible individuals with respect to each constraint for two terrains

ure 7. For easier terrains, smaller number of control points is enough to provide better paths. As number of control points increase, the path to route becomes more complex and especially the length of the path increases which results in worse fitness values.

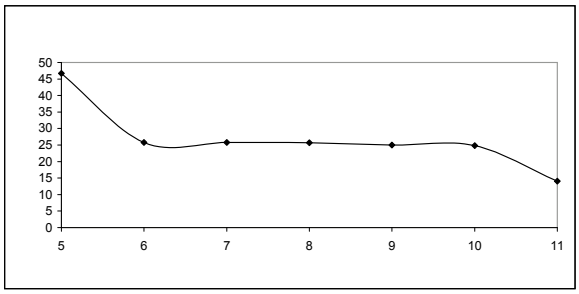

Figure 7: Fitness value of the best solution by varying the number of control points for terrain 1

However, as terrain structure becomes harder (i.e there are large number of hills with multiple peaks), more control points are necessary to deal with the difficulty. Better results are observed in our experiments for larger number of control points. Generally, individuals which have 7-8 control points give the best fitness in the population; while less or more control points may results in fitness decline for difficult or complex terrains. For terrain 2, the effect of the number of control points on the fitness value is shown in figure 8.

As mentioned in Section 3.3, although three different mutation operators are proposed in this study, it is clear that some of them (such as deletion operator) may not be applied alone. Therefore, we consider following four different test cases by using these three operators.

- *Update mutation.* In this case, we consider only update mutation for all individuals at every generation.

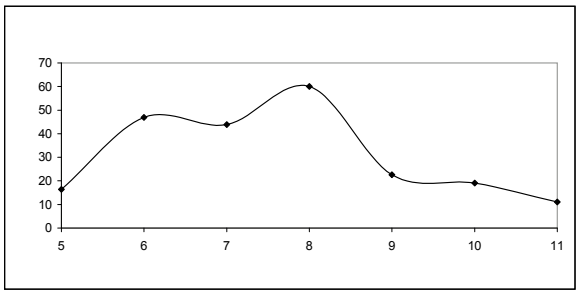

Figure 8: Fitness value of the best solution by varying the number of control points for terrain 2

- *Random insert/delete mutation.* In this case, the insertion or deletion operator is selected randomly for all individuals at every generation. Therefore, consecutive insertions or deletions can be possible.
- *Insert/delete mutation with a threshold value.* In this case, the operation type is decided according to the number of control points the individual has. If the individual has number of control points less than or equal to a specified threshold value, then the insertion mutation is applied, else if the individual has number of control points more than the threshold value then the deletion mutation is applied. Thus, it's not possible to have too many control points, since it generally increases the cost; or it is not possible to have too little number of control points, since it does not provide sufficient representation of the individual.
- *Random insert/delete/update mutation.* The second test case is extended with the update mutation operator.

The number of control points is set to 5 for the first terrain; while best fitness value is set with 8 control points for the terrain 2. In this set of the experiments, number of generation is set to 1000; and each experiments is repeated 10 times with different seeds. Minimum normalized cost values (out of 10 runs) and average normalized costs of each mutation type for both terrain 1 and terrain 2 are given in Tables 1 and 2, respectively.

Table 1: Normalized cost values for different mutation types for terrain 1

Mutation Type	Minimum Cost	Average Cost
update	0.001161	0.081276
random insert/delete	0	0.096683
smart insert/delete	0	0.085790
random insert/delete/update	0	0.110593

Figure 9 and Figure 10 show sample paths from source to destination for terrain 1 and terrain 2 given in Figure 2, respectively.

5. CONCLUSION

In this paper, we present a 3D path planner for the navigation of Unmanned Aerial Vehicles (UAVs) by using evolutionary algorithms. The flight path (represented by a

Table 2: Normalized cost values for different mutation types for terrain 2

Mutation Type	Minimum Cost	Average Cost
update	0.004935	0.176629
random insert/delete	0	0.125249
smart insert/delete	0.001126	0.104528
random insert/delete/update	0	0.112339

Figure 9: Sample path for Terrain 1

parametrized B-Spline curve) is generated by considering both maneuver and technical capabilities of the UAVs. We are in the process of extending this study by including two additional mission specific flights, which are sweeping flights and flight envelopes in a given line of sight.

6. REFERENCES

[1] UAV 2003:A Roadmap for Deploying Unmanned Aerial Vehicles (UAVs) in Transportation, Specialist Workshop, December 2003, U.S. Department of Transportation.

[2] Stefano Caselli, Monica Reggiani, Roberto Rocchi, Heuristic Methods for Randomized Path Planning in Potential Fields, Proceedings of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, pp. 426-431, 2001.

[3] Yao-hong Qu, Quan Pan, Jian-guo Yan, Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms, IECON 2005, Industrial Electronics Society, 31st Annual Conference of IEEE 2005.

[4] Ioannis K. Nikolos, Athina N. Brintaki, Coordinated UAV Path Planning Using Differential Evolution, Proceedings of the 13th Mediterranean Conference on Control and Automation, pp. 549-556, 2005.

[5] Tom Schouwenaars, Jonathan How, and Eric Feron, Receding Horizon Path Planning with Implicit Safety Guarantees, Proceeding of the 2004 American Control Conference, vol. 6, pp. 5576-5581, 2004.

[6] H. Burchardt, R. Salomon, Implementation of Path Planning using Genetic Algorithms on Mobile Robots, Congress on Evolutionary Computation, CEC 2006, pp. 1831-1836, 2006.

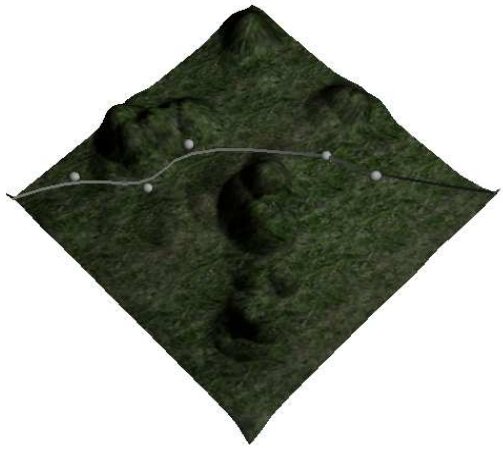

Figure 10: Sample path for Terrain 2

[7] Yanrong Hu, Simon X. Yang, A Knowledge Based Genetic Algorithm for Path Planning of a Mobile Robot, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4350-4355, 2004.

[8] Guo Tong-ying, Qu Dao-kui, Dong Zai-li, Research of Path Planning for Polishing Robot Based on Improved Genetic Algorithm, Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics, pp. 334-338, 2004.

[9] D. Rathbun, S. Kragelund, A. Pongpunwattana, B. Capozzi, An Evolution Based Path Planning Algorithm For Autonomous Motion of a UAV Through Uncertain Environments, Proceedings of Digital Avionics Systems Conference, vol. 2, pp. 8D2-1-8D2-12, 2002.

[10] I. K. Nikolos, N. Tsourveloudis, and K. P. Valavanis, Evolutionary algorithm based 3-D path planner for UAV navigation, Proceedings of the 9th Mediterranean Conf. on Control and Automation, 2001.

[11] jME graphics API: http://www.jmonkeyengine.com/

[12] Jason Shankel (Maxis) Fractal Terrain Generation - Midpoint Displacement, Game Programming Gems, pp. 503-507, 2000.

[13] Terrain Generation Tutorial: Hill Algorithm, http://www.robot-frog.com/3d/hills/hill.html

[14] Ioannis K. Nikolos, Kimon P. Valavanis, Nikos C. Tsourveloudis, Anargyros N. Kostaras, Evolutionary Algorithm based Offline/Online Path Planner for UAV Navigation, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, vol.33, no. 6, pp. 898-912, 2003.

[15] A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing, Springer (Natural Computing Series), 2003.

[16] Gerald Farin, Curves and Surfaces for CAGD (Computer Aided Graphics and Design): A Practical Guide, Fifth Edition, Morgan Kaufmann, 2001.