Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation, and Mathematics

PROCEEDINGS

of the 18th International Conference on Process Control

Hotel Titris, Tatranská Lomnica, Slovakia, June 14 – 17, 2011

ISBN 978-80-227-3517-9

http://www.kirp.chtf.stuba.sk/pc11

Editors: M. Fikar and M. Kvasnica

Al-Rashedi, N., Gerke, M.: 3D Off-Line Path Planning for Autonomous Airships in Restricted Known Environments., Editors: Fikar, M., Kvasnica, M., In *Proceedings of the 18th International Conference on Process Control*, Tatranská Lomnica, Slovakia, 182–187, 2011.

Full paper online: http://www.kirp.chtf.stuba.sk/pc11/data/abstracts/071.html

3D Off-Line Path Planning for Autonomous Airships in Restricted Known Environments.

Naef Al-Rashedi and Michael Gerke

Fakultät für Mathematik und Informatik
Lehrgebiet Prozesssteuerung und Regelungstechnik
Fernuniversität in Hagen
Universitätstrasse 27, D-58097 Hagen.
Tel: +49 23 31 / 9 87 – 17 02
Fax: +49 23 31 /9 87 – 3 54
Naef.Al-Rashedi@FernUni-Hagen.de

Abstract: The paper presents a Genetic Algorithm (G.A.) for off-line path planning of Autonomous Small Airships in known 3D environments with special consideration of restricted areas. The algorithm assumes that the airship is used in Fire Fighting and Mine Detection projects, so the aircraft will fly only a few meters above the ground, which means there is a high possibility of collision with obstacles. The task of the Off-Line Path Planner algorithm is to find an optimal route to visit all the predefined locations for airborne measurement exactly once per mission, without any collisions with environmental obstacles and to avoid fly over a defined restricted area. The planner task posed here is an NP problem. This paper proposes a 3D Off-Line Path Planner using G.A. including chromosome representation, G.A. crossover and collision avoidance with known obstacles. The proposed algorithm is implemented using MATLAB with Genetic Algorithms and Mapping Toolboxes. The proposed algorithm is tested using real maps of our research airfield and the result shows that the algorithm finds a near-optimal collision free path for the airship.

1. Introduction

Autonomous Small Airships have many applications in civil and military areas[3], e.g. for different missions including fire fighting, mine detection, traffic surveillance, maintenance of high power electric lines and advertising. The main advantages of using an airship are: it has low energy consumption, there is no vital risk for operators during performance of hazardous missions, and its long endurance in the air.

The airship need a control system that can make both low-level control decisions in real-time, medium-level decisions such as path planning, and high-level decisions, such as cooperative task assignment, for long time missions without human interference. Task assignment is crucial for designing successful missions in difficult environments while the path planners which generate collision-free and optimized paths are needed to give autonomous operation capability to the airship. The combined solution of both aspects for the mission planning problem leads to a near optimal flight trajectory [6].

Figure (1): Fernuniversität Airship

The paper is organized as follows. Section 2 formulates the path planning problem. The description of the proposed path planner follows in section 3. In section 4 experiments and results are given. The conclusion of this paper is drawn in section 5.

2. Problem Definition

The mobile robot path planning problem is typically formulated as follows: given a mobile robot and a description of an environment and set of user-defined way-points (wp), the problem is to calculate a route that visits all user-defined way-points exactly once. The resulting path should be free of collision and satisfy certain optimization criteria (i.e., shortest path)[3]. This problem corresponds to finding the shortest Hamiltonian cycle in a complete graph G = (V, E) of an ***n*** nodes. Thus the path planner consists of finding a permutation of the set $\{wp_1, wp_2, wp_3, \ldots, wp_N\}$ that minimize the quantity:

$$\text{Minimize} \quad \left[\sum_{i=1}^{N-1} d\left(wp_i, wp_{i+1}\right) + d\left(wp_N, wp_1\right)\right]$$

Where $d(wp_i, wp_j)$ denotes the distance between waypoint wp_i and waypoint wp_j.

Researchers distinguish between various methods used to solve the path planning problem according to two factors, (1) the environment type (i.e., static or dynamic), (2) the path planning algorithms (i.e., Off-Line or On-Line). A static environment is defined as the environment which doesn't contain any moving objects other than a navigating robot; while any dynamic environment includes moving objects (i.e., human beings, vehicles and other robots).
The Off-Line path planning algorithm requires a complete knowledge about the search environment and is based on the fact that all terrain should be static. On the other hand, On-Line path planning means that the path planning algorithm is calculated in real-time while the robot is moving around. In other words, the algorithm is capable of producing a new path in response to environmental changes [2]. Many studies try to solve the path planning problem for mobile robotics by using evolutionary approaches like genetic algorithms [1,5,7,10], but most of them solve the problem as how to go from one location to another and this study try to find an optimal trajectory to visit many locations exactly once with respect of collision avoidance, also this study is different from [9] by introducing the existence of user-defined restricted area that make the problem of path planning more complicated.

3. The Proposed Path Planner

The proposed path planner accomplishes its task into two phases: the preparation phase and the G.A.

3.1. The Preparation Phase

In this phase all inputs data such as digital 3D maps and user-defined way-points are represented and prepared to be use by the genetic algorithm.

3.1.1 Environment Representation

This paper considers that the airship will fly in static and well known environments, while these environments are represented as digital maps (e.g. for processing in MATLAB Mapping Tool Box).

3.1.2 Restricted Area Avoidance

The procedure starts with checking if the direct (straight line) path between any two of the User-Defined locations is crossing a defined restricted area or not. If it is crossing a restricted area then, a process starts to find new subway points to avoid this area. There are only two possible ways to avoid the restricted area, one is to go left way side and the other is to go right way, the decision of where to go (left or right) is taken according which one is the short. The technique used to determine the short way to avoid the restricted area can be described as the following: suppose that (UDP_i, UDP_j) are two User-Define points, and $\{v_1,v_2,v_3,v_4,v_5\}$ are set of vertices of a restricted area as shown in figure(3). The direct path from UDP_i to UDP_j is crossing the restricted area and divide its vertices into two sets: the vertices $\{v_1,v_2\}$ that are located to the right of direct path line, and the vertices $\{v_3,v_4,v_5\}$ that are located to the left. The alternative path to avoid the restricted area in this example can be defined as:

Minimum(|(UDP_i,v_1)+(v_1,v_2)+(v_2,UDP_j)|,
|(UDP_i,v_3)+(v_3,v_4)+(v_4,v_5)+(v_5,UDP_j)|)

It is clear from figure(3) that |(UDP_i,v_1)+(v_1,v_2)+(v_2,UDP_j)| is the minimum in length so, the vertices $\{v_1,v_2\}$ are added as new subway points into the path between (UDP_i and UDP_j) to avoid the restricted area. Another issue in restricted area avoidance is how to determine the elevation of the new subway points? The elevations of UDP_i and UDP_j are defined by user and the elevation of the new subway points can be defined as:

Maximum ($eUDP_i$, $eUDP_j$, eh)

Where: $eUDP_i$,$eUDP_j$ are the elevations of UDP_i,UDP_j in order, and **eh** is the elevation of the highest point of the terrain in the path [UDP_i,v_1,v_2,UDP_i] .

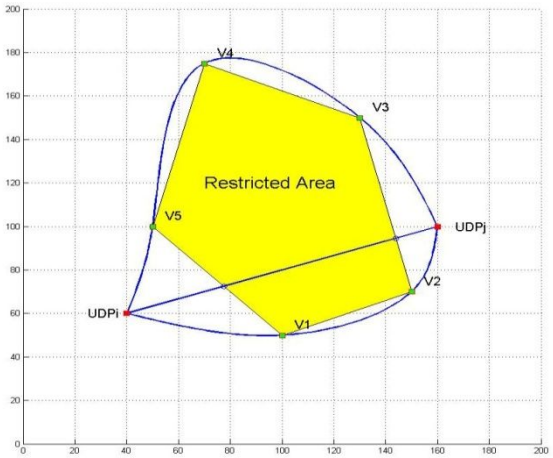
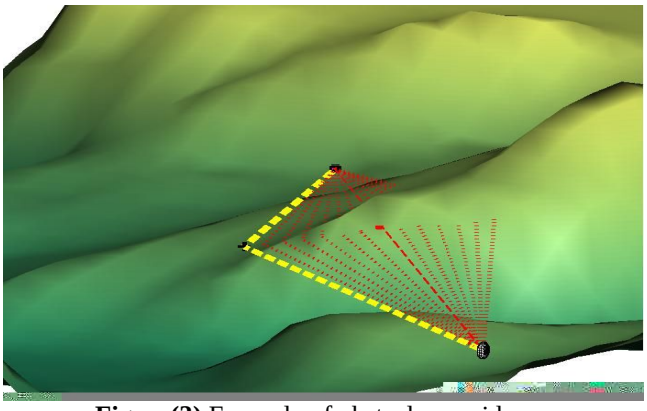


Figure (3): Example of restriced area avoidence

3.1.3. Verifying Feasibility and Adding New Subway Points

The Feasibility of all direct paths between each set of user-defined way-point are verified and if there is a feasible direct path between any pair of these points, its path length is calculated and stored in a cost table, otherwise (i.e. there is an obstacles between two waypoints), a process is being started to find an indirect feasible path between these two user-defined way-points by adding a new subway point



Figure(3) Example of obstacles avoidance

to avoid this disturbing obstacle. The strategy of adding new subway points is based on making a displacement around the obstacle in all directions in order to find an intermediate way-point that is feasible from both user-defined way-points under consideration. The main advantage of this step is to reduce the calculation and the execution time of G.A.

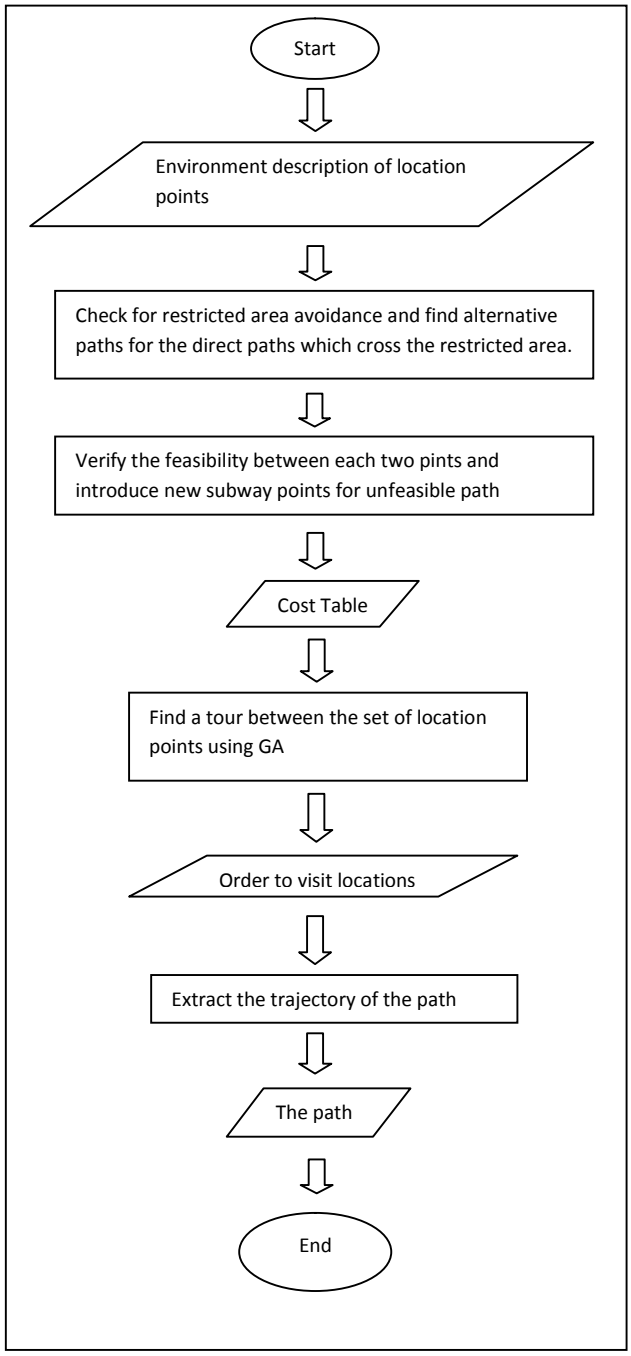


Figure (4): Path Planner Process Flowchart

3.2. The G.A. Algorithm:

This new pre-processing strategy enhances the G.A. search that is used to solve the travel salesman problem in [8] to be used as an optimization tool for the airship path planner. A description of the G.A. search for 3D-problems is as follows:

3.2.1 Path Representation:

The proposed algorithm represents a robot mission as an Upper Triangle Binary Matrix (UTBM) as in **Figure (5)** which represents the tour (1, 3, 5, 2, 4, 6). Every chromosome is represented in binary form, which means: if the element (i,j) of the matrix is equal (1) there is a feasible path between user-defined way-points (i) and (j) in the tour.

	2	3	4	5	6
1	0	1	0	0	1
2	0	1	1	0	
3	0	1	0		
4	0	1			
5	0				

Figure (5): The Proposed Representation of a Chromosome

The matrix representation must satisfy the following conditions in order to represent a feasible tour:

1- The number of elements in the matrix that have the value (1) must be equal to the number of the user-defined way-points.

$$\sum_{i=0}^{N-2} \sum_{j=1}^{N-1} wp_{ij} = N$$

2- The number of matrix elements that have the value of (1) in each row and each column of the same way-point must be equal to two.

$$Col_j = \sum_{i=0}^{N-2} c_{ij}$$

$$Row_i = \sum_{j=1}^{N-1} c_{ij}$$

$$\forall (i = j): Row_i + Col_j = 2$$

3.2.2. The Crossover

The crossover operation generates an offspring from two parents as follows:

1- Unify the two parents matrices in one matrix by executing (Or) operation.
2- The result matrix from step 1 may be an invalid tour (do not satisfy the two conditions from above). So it must be repaired by counting the number of edges of each way point. If any way point has more than two edges, then keep the shortest two edges and delete the others. The way points that have less than two edges are added to a list of disconnected way points.
3- Adding the missing edges to disconnected way points by using the greedy algorithm.

3.2.3. The Evaluation Function:

The evaluation function is an important part of any evolutionary process using G.A. Only an appropriate selection of the evaluation function will lead the search towards the desired optimal solution. For the airship that is used in complex flight missions, the optimal path has two type of constrains:

- Hard constrains: the path should be free of collision and every user-defined way- point of the flight mission should be visited exactly once. (The algorithm proposed here guarantees that all the solutions that are produced meet these conditions).
- Soft constrains which can be defined by user requirements: in specific scenarios such as fire fighting and mine detection some additional constrains can be formulated, e.g. enforcement of a shortest path and of path types keeping the airship in low altitude above the ground. The following equation is used as evaluation function for the algorithm proposed here:

$$\mathbf{F = minimize\ [\ |T| + H\ ]}$$

Where:

Parameter **|T|** is the length of the flight mission [e.g. in meters] and is computed as the sum of the paths lengths between every two way-points of the tour.

Parameter **H** is a value computed as following:

$$\mathbf{H = h * Hf}$$

Where:

h is the sum of the difference between the maximum height of the airship during the tour and the height of the user-defined way-points.

Hf is a user defined value; for a large value the GA will select the solution that avoids the airship to go up even if the tour will be longer.

4. Experiments and Results

The proposed algorithm is implemented using MATLAB and tested both in a real three-dimensional map of our experimental area in Hemer (Northrhine-Westfalia), and sample Matlab DEM files like Geoid and South San Francisco. The following figures are showing the results of the experiments.

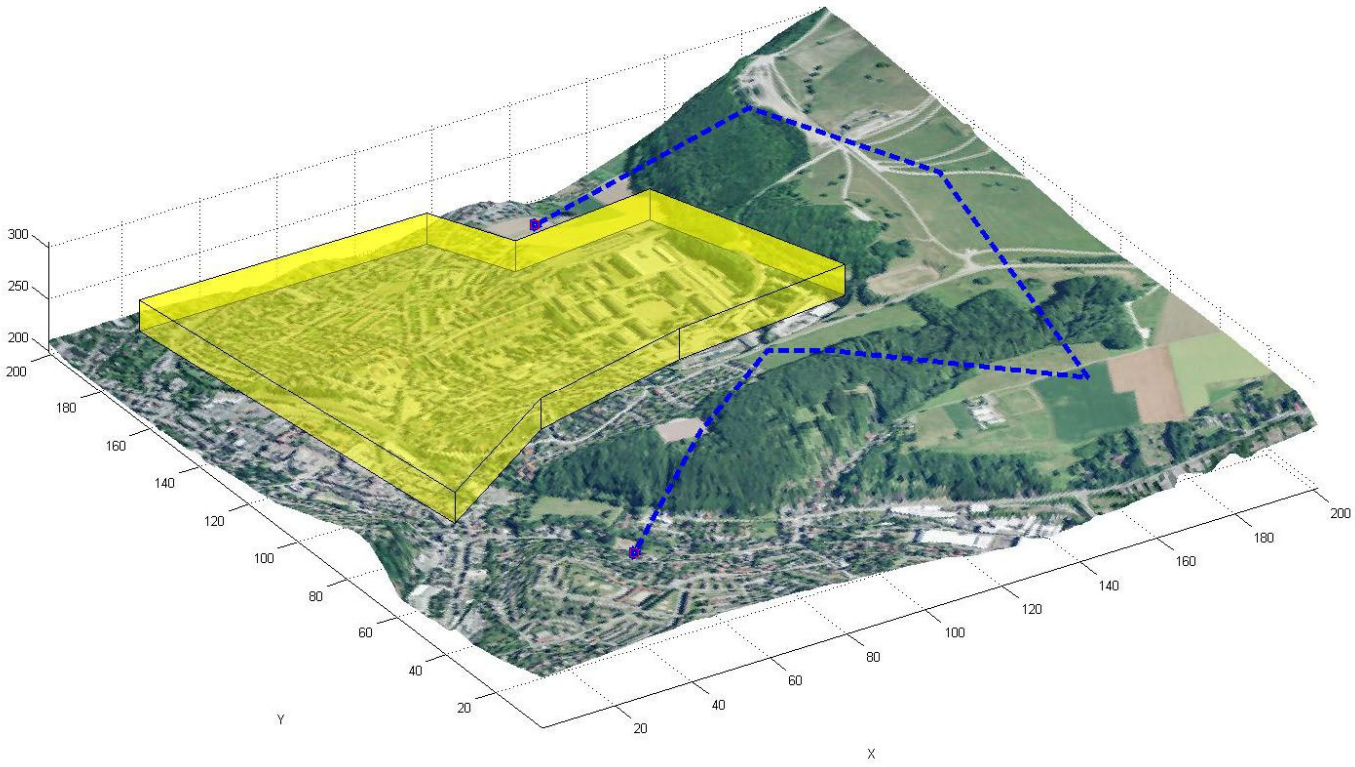


Figure (6): Sample tour for 6 user defined way-points in map of Hemer.

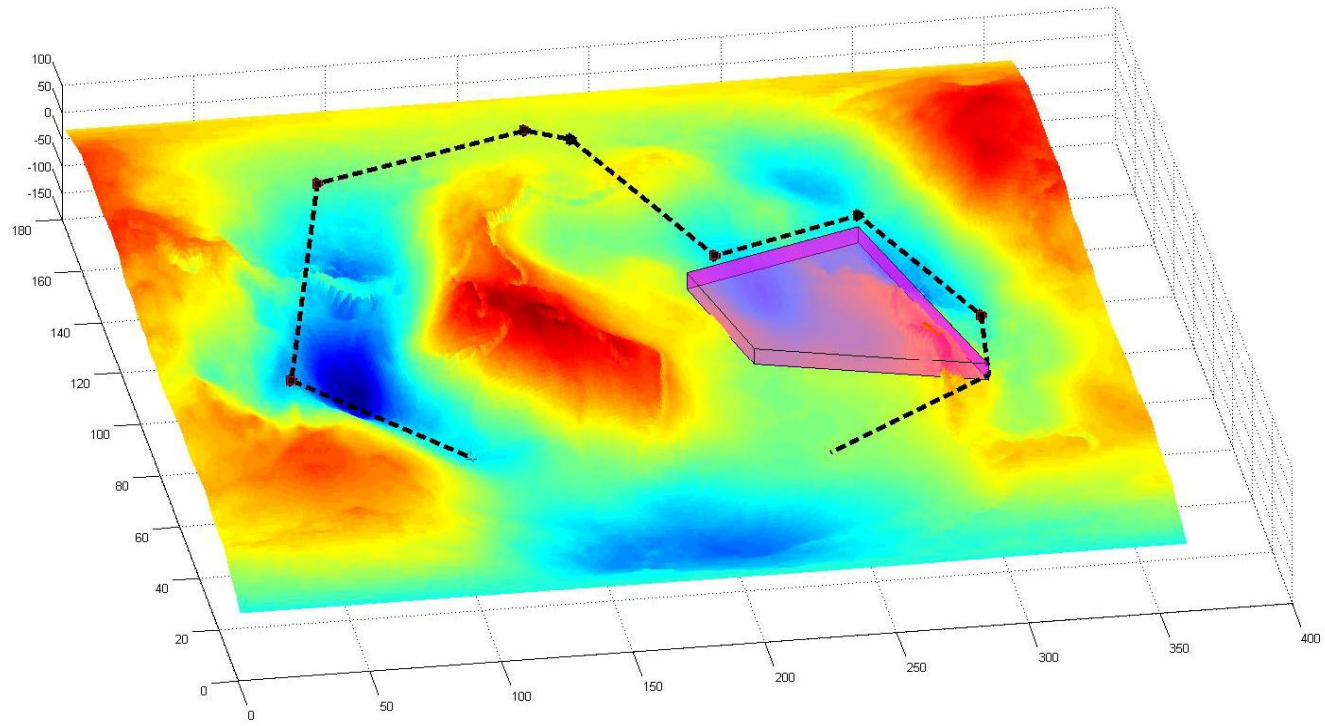


Figure (7): A sample tour of 7 user defined way-points in map Geoid

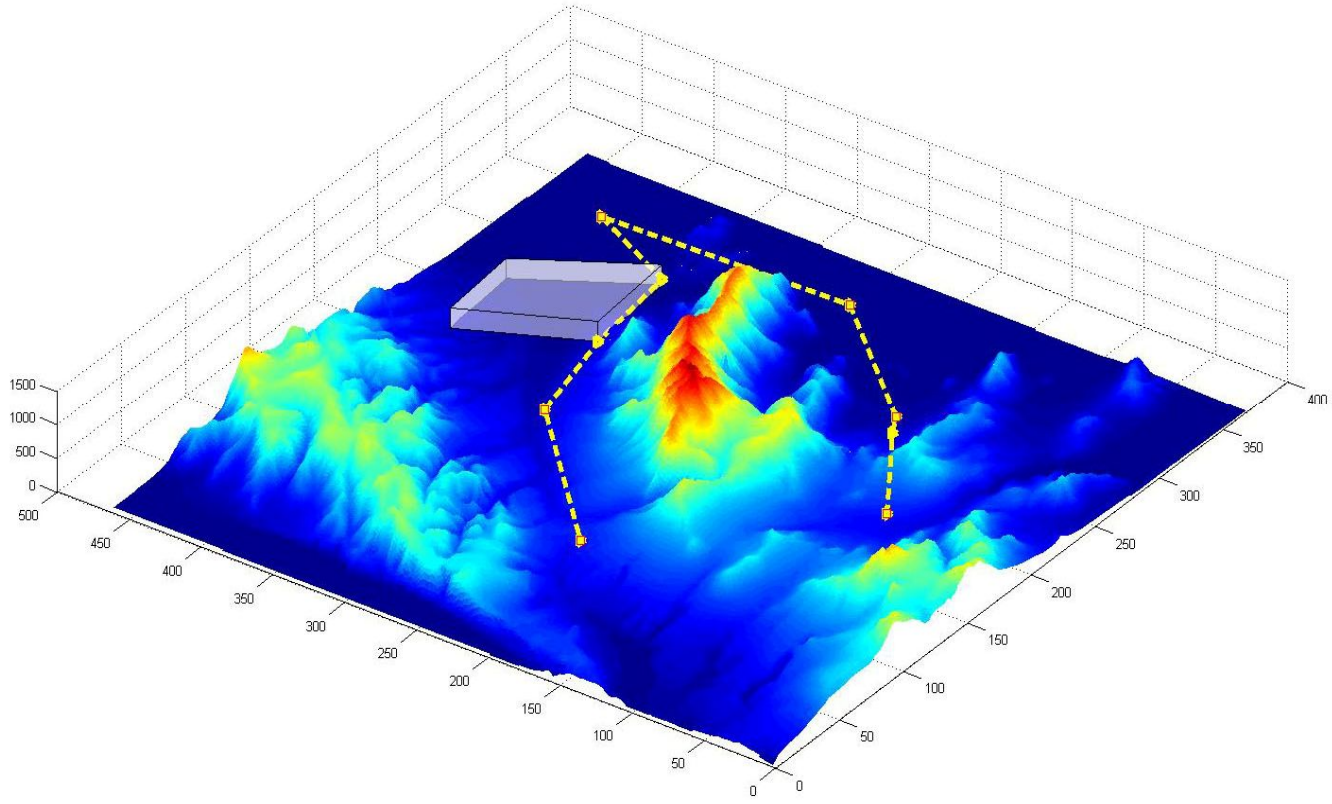


Figure (8): Sample tour of 6 User-Define points in South San Francisco map.

5. Conclusion:

This paper represents an Off-Line path planner for small autonomously operating airships in restricted but well known static 3D environments. Using G.A. search, the separation between Preparation phase and the G.A. process reduces the execution time of G.A. and makes the path planner more flexible. For example, it is relatively simple to modify the proposed G.A., or to integrate optimization criteria considering further factors of influence (like the direction of wind force, the energy consumed by airship devices …etc.). Another advantage of using GA as problem solver is, that in case of any changes in the environment (e.g. the boundaries of the restricted area are change.) the last solution of the GA planner is still important and it can be used as initial solution for an updated search procedure. This speeds-up the calculation of an optimal solution with respect to recent changes in the environment and it makes the GA path planner more efficient for on-line operation during the airship mission in a dynamically changing environment.

6. References

[1]. Anargyros N. Kostaras, Ioannis K. Nikolos, Nikos C. Tsourveloudis, and Kimon P.Valavanis: *Evolutionary Algorithm Based Off-Line/On-Line Path Planner for UAV Navigation*. Proceedings of the 10th Mediterranean Conference on Control and Automation – MED 2002 Lisbon, Portugal, July 9-12. 2002.

[2]. Bruno Siciliano, Oussama Khatib, Frans Groen: *Robot Navigation from Nature Simultaneous Localisation, Mapping, and Path Planning Based on Hippocampal Models.* Springer Tracts in Advanced Robotics Volume 41 ISSN 1610-7438. Springer-Verlag Berlin Heidelberg.2008

[3]. Christopher Bolkcom, *Potential Military Use of Airships and Aerostats*, CRS Report for Congress, USA, 2004

[4]. Florian Adolf, Franz Andert, Sven Lorenz, Lukas Goormann, and Jörg Dittrich: *An Unmanned Helicopter for Autonomous Flights in Urban Terrain.* German Workshop on Robotics GWR2009, Braunschwrig-Germany, 2009.

[5]. Francois C.J. Allaire, Mohamed Tarbouchi, Gilles Labonte and Giovanni Fusina: *FPGA Implementation of Genetic Algorithm for UAV Real-Time Path Planning.* Springer Science. 2008

[6]. Isil Hasirciolgu Haluk Rahmi Murat Ermis: *3-D Path Planning for the Navigation of Unmanned Aerial Vehicles by Using Evolutionary Algorithms*, GECCO'08, Atlanta-Georgia USA, 2008

[7]. Ismail Al-Taharwa, Alaa Sheta and Mohammed Al-Weshah: *A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment*, Journal of Computer Science 4 (4): 341-344, ISSN 1549-3636, Science Publication, 2008.

[8]. Naef Taher Al-Rashedi and Jalal Atoum: *Solving Travel salesman Problem Using New Operator in Genetic Algorithms*, American Journals of Applied Sciences 6 (8): 1586-1590, 2009.

[9]. Naef Al-Rashedi and Michael Gerke, *3D Off-Line Path Planning for Autonomous Airships in Known Environments by using Genetic Algorithms.* Proceeding 20. Workshop Computational Intelligence, Dortmund Germany, 1-3 December 2010.

[10]. Zixing, CAI., Lingli, YU., Chang, XIAO., Lijue, LIU.: *Path Planning for Mobile Robots in Irregular Environment Using Immune Evolutionary Algorithm*. Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea, July 6-11, 2008.