Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation

Ioannis K. Nikolos, Kimon P. Valavanis, *Senior Member, IEEE*, Nikos C. Tsourveloudis, and Anargyros N. Kostaras

***Abstract*—An evolutionary algorithm based framework, a combination of modified breeder genetic algorithms incorporating characteristics of classic genetic algorithms, is utilized to design an offline/online path planner for unmanned aerial vehicles (UAVs) autonomous navigation. The path planner calculates a curved path line with desired characteristics in a three–dimensional (3-D) rough terrain environment, represented using B-Spline curves, with the coordinates of its control points being the evolutionary algorithm artificial chromosome genes.**

Given a 3-D rough environment and assuming flight envelope restrictions, two problems are solved: i) UAV navigation using an offline planner in a known environment, and, ii) UAV navigation using an online planner in a completely unknown environment. The offline planner produces a single B-Spline curve that connects the starting and target points with a predefined initial direction. The online planner, based on the offline one, is given on-board radar readings which gradually produces a smooth 3-D trajectory aiming at reaching a predetermined target in an unknown environment; the produced trajectory consists of smaller B-Spline curves smoothly connected with each other. Both planners have been tested under different scenarios, and they have been proven effective in guiding an UAV to its final destination, providing near-optimal curved paths quickly and efficiently.

***Index Terms*—3-D path planning, B-splines, evolutionary algorithms, navigation, UAV.**

I. Introduction

THIS work has been motivated by the challenge to develop and implement a single path planner for autonomous unmanned aerial vehicle (UAV) navigation and collision avoidance in known, completely unknown, and/or partially known 3-D rough terrain environments.

The problem being solved considers UAV flight envelope restrictions in terms of enforced maximum flight height and minimum turning radius, obstacles being the 3-D rough terrain (mountains, valleys, etc.), as well as moving ones. The UAV is assumed to be equipped with a set of on-board sensors, including radar, global positioning system (GPS), differential GPS (DGPS), inertial navigation system (INS), and gyroscopes, through which it can sense its surroundings and position. The final destination (end-point coordinates) is known and the UAV must follow an as smooth as possible trajectory (imitating real flight restrictions), planned and re-planned in real-time, avoiding static (mountains), and moving obstacles given its initial position and initial flight direction. The vehicle is assumed to be a point (its actual size is taken into account by equivalent obstacle—ground growing). Therefore, two problems are being solved.

1) *UAV Navigation Using an Offline Planner Considering a Known 3-D Environment:* The offline planner generates collision free paths in environments with known characteristics and flight restrictions (acquired via 3-D GIS based generated maps or otherwise). The derived path line is a single continuous 3-D B-Spline curve, while the solid boundaries are interpreted as 3-D rough surfaces. The air vehicle course is actually a curve with curvature continuity that cannot be modeled using straight-line segments (which is the usual practice for ground robots). Therefore, B-Spline curves based path representation is utilized having the advantage of being described using a small amount of data (the coordinates of their control points), although possibly producing very complicated curves.

2) *UAV Navigation Using an Online Planner Considering a Completely Unknown 3-D Environment:* The online planner uses acquired information from the UAV on-board sensors (that scan the area within a certain range from the UAV). The online controller rapidly generates a near optimum path that will guide the vehicle safely to an intermediate position within the sensors' range. The process is repeated until the final position is reached. The path line from the starting point to the final goal is a smooth, continuous 3-D line that consists of successive B-Spline curves, smoothly connected to each other.

The UAV path-planning problem is considered within an evolutionary algorithm (EA) context, in which the path line is represented using B-Spline curves, with the coordinates of its control points being the EAs artificial chromosome genes. The reasons behind choosing EAs as an optimization tool for the path-planning problem are their high robustness compared to other existing directed search methods, their ease of implementation in problems with a relatively high number of constraints, and their high adaptability to the special characteristics of the problem under consideration.

The paper contributions include the development of an EA based offline/online path planner, suitable for UAV navigation in both known and completely unknown environments and the construction of smooth and easily followed path lines. The path line in the online procedure is gradually constructed, using on board sensors information. Additionally, a potential field is utilized that drives the path line to bypass obstacles

Manuscript received April 4, 2002. This work was supported in part by two research grants from the Greek Secretariat of Research and Technology PAVE99-BE118 and PAVET00-BE148. This paper was recommended by Associate Editor H. Takagi.

I. K. Nikolos, K. P. Valavanis, and N. C. Tsourveloudis are with the Technical University of Crete, Department of Production Engineering and Management, Chania 73100 Greece (e-mail: jnikolo@dpem.tuc.gr, kimonv@dpem.tuc.gr, nikost@dpem.tuc.gr).

A. N. Kostaras is with the London School of Economics, London, WC2A 2AR, U.K. (e-mail: A.Kostaras@lse.ac.uk).

Digital Object Identifier 10.1109/TSMCB.2002.804370

lying between the UAV and its final destination. Further, an additional EA-based procedure is introduced that forces the UAV to bypass concave obstacles and avoid local optima.

A. Related Work

There already exist methods that produce either 3-D, or 2-D trajectories for guiding mobile robots in known, unknown, or partially known environments. In some cases, neural network based controllers were designed and trained to guide a robot when unknown static obstacles were sensed [1]. In other cases, fuzzy based controllers were used to solve the 2-D mobile robot online navigation problem, with its parameters being optimized in real time, through an evolutionary procedure, such as in [2].

During the past few years, it has been shown by many researchers that EAs are a viable candidate to solve such problems, including the path planning problem, effectively and provide feasible solutions within a short time without demanding excessive computer power. Traditionally, EAs have been used for the solution of the path-finding problem in ground based or sea surface navigation [3]. Commonly, the generated trajectory had the form of a crooked line that guided a mobile robot or a vehicle along a 2-D path on the earth's surface or the sea surface. The genes used, represented the path point coordinates the vehicle changes its direction to. Other approaches took into account the time dimension by using genes that also described the vehicle steady speed as it traversed a part of its path. When the vehicle's operational environment was partially known or dynamic, a feasible and safe trajectory was planned offline by the EA, and the algorithm was used online whenever unexpected obstacles were sensed [4], [5].

EAs (with binary coding) have also been used for solving the path-finding problem in a 3-D environment for underwater vehicles, assuming that the path is a sequence of cells in a 3-D grid [6], [7]. In addition, B-Spline curves have been used for trajectory representation in 2-D environments (simulated annealing based path line optimization, combined with fuzzy logic controller for path tracking) [8], and in 3-D environments (evolutionary algorithm based path line optimization for a UAV over rough terrain) [9]. A 3-D heuristics-based planner has been presented in [10] for the local trajectory planning in a partially known environment with moving obstacles but with predefined global path (in the form of knot points with known coordinates). Other related work for the 2-D case, may be found in [13]–[15].

The current work implements the EAs in the demanding environment of UAV flight over a rough, completely unknown ground surface, without considering a predefined path. Contrary to the ground based, sea surface, or underwater vehicles, UAV flight wrong decisions and strategies may easily result in destroying the UAV. For this reason, the feasibility of the path line is the main concern, while special procedures are used to overcome navigation problems with concave obstacles. Nevertheless, the high velocities and the flight dynamics impose specific constraints for the smoothness and the curvature of the calculated path line, leading to the adoption of the B-Spline formulation. It is emphasized that the use of straight line segments, or a sequence of cells for the construction of the path line (as it is the practice for surface and underwater 3-D environments) is not applicable in this case (UAV flight), due to flight envelope restrictions and stability problems. For the same reasons, gradually constructed B-Spline curves are adopted in the online planner. The radar range provides the time needed for calculating smooth near optimal curves connected to each other, that are naturally fitted to the UAV flight limitations.

The rest of the paper is organized as follows: Section II summarizes EA fundamentals along with the basic features of B-Spline curves. Specific EA features as applied to the problem under consideration are presented. The offline planner is presented in Section III, followed by the presentation of the online planner in Section IV. Experimental results are shown in Section V, followed by discussion and conclusions in Section VI. The algorithms developed for the online planner are included in Appendix A and the basic equations for the construction of 3-D B-Spline curves in Appendix B. Finally, the EA parameters selection procedure is presented in Appendix C.

II. Fundamentals of Evolutionary Algorithms

EAs are a class of search methods with remarkable balance between exploitation of the best solutions and exploration of the search space. They combine elements of directed and stochastic search and, therefore, are more robust than existing directed search methods [3]. Additionally, they may be easily tailored to the specific application of interest, taking into account the special characteristics of the problem under consideration.

The natural selection process is simulated in EAs, using a number (population) of individuals (solutions to the problem) to evolve through certain procedures. Each individual is represented through a string of numbers (bit strings, integers, or floating point numbers) in a similar way with chromosomes in nature. Each individual's quality is represented by a fitness function tailored to the problem to be optimized.

Classical EAs use binary coding for the representation of the genotype [3]. However, floating point coding moves the EAs closer to the problem space, allowing the operators to be more problem specific. For this reason floating point coding is used in the current work, which provides a better physical representation of the path line control points and easier control of the space constraints. Additionally, two points that are close to the physical space are also close in the representation space (the genotype encoding), and vice versa. With this type of encoding directed search techniques gain physical representation and they are easily applicable.

The EA starts by generating, randomly, the initial chromosome population with their genes taking values inside the desired constrained space. After the evaluation of each individual's fitness function, operators are applied to the population, simulating the according natural processes. Applied operators include various forms of selection, recombination, and mutation, which are used in order to provide the next generation chromosomes. The process of a new generation evaluation and creation is successively repeated, providing individuals with high values of fitness function. Each chromosome consists of the same (fixed) number of genes (for the problem at hand, they are the coordinates of B-Spline control points).

The first operator applied to the selected chromosomes is the classical one-point crossover scheme [3]. Two randomly selected chromosomes are divided in the same (random) position, while the first part of the first one is connected to the second part of the second one, and vice-versa. The crossover operator is used in order to provide information exchange between different potential solutions to the problem.

The second operator applied to the selected chromosomes is the classic uniform mutation scheme. This asexual operator alters a randomly selected gene of a chromosome. The new gene takes its random value from the constrained space, determined in the beginning of the process (in the offline planner being the borders of the physical 3-D search space). The mutation operator is used in order to introduce some extra variability into the population.

In order to provide fine local tuning, nonuniform mutation and heuristic crossover are used, along with the classic mutation and crossover schemes [3]. The first operator chooses randomly, with a predefined probability, the gene of a chromosome to be mutated. Contrary to the uniform mutation described above, the search space for the new gene is not fixed, but it shrinks close to the previous value of the corresponding gene as the algorithm converges. The search is uniform initially, but very local at later stages.

The second operator generates a single offspring x_3 from two parents x_1 and x_2. If x_2 is not worse than x_1, then x_3 is given as

$$x_3 = r(x_2 - x_1) + x_2 \tag{1}$$

where r is a random number between 0 and 1. In this way a search direction is adopted, providing fine local tuning and search in the most promising direction (keep in mind that the chromosome genes are physical coordinates and the direction of search actually has a physical meaning). The two last operators (especially the heuristic crossover operator), proved to be critical in obtaining a high convergence rate and a feasible solution in a few generations (see Appendix C).

The initial population is created randomly in the constrained space of each gene. The lower and higher constraints of each gene may be chosen in a way that specific undesirable solutions may be avoided, such as path lines with a higher than desired altitude. Although the shortening of the search space reduces the computation time, it may also lead to sub-optimal paths, due to the lower variability between the potential solutions.

The EA discussed in this work is a modified breeder genetic algorithm (BGA) incorporating some characteristics of the classic genetic algorithms (GAs). Breeder genetic algorithms use floating-point representation of variables and both recombination and mutation operators. The truncation model is used as the selection scheme, with the best $T\%$ of P initial individuals to give origin to the individuals of the next generation, with equal probability.

The selection scheme used is hybrid, a combination of the truncation model of BGAs and the roulette procedure of traditional GAs. Starting with the truncation model, only $T\%$ elements, showing the best fitness, are chosen in order to give origin to the individuals of the next generation, with parameter T being the *threshold* of the procedure. Once chosen, these individuals are used for the generation of a new population through the classic roulette wheel selection [3]. An elitist model assures that the best individual of each generation always survives the selection procedure and reproduces its structure in the next generation.

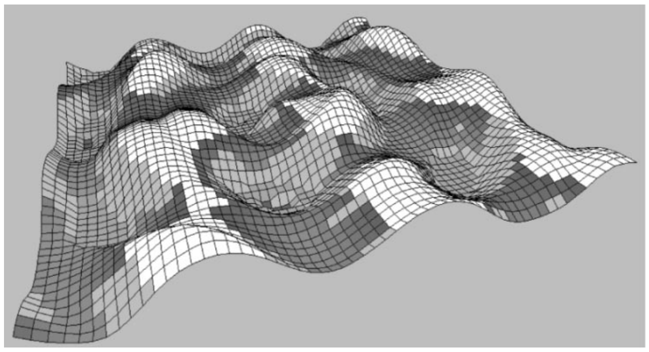
Fig. 1. Artificial terrain produced using (2).

This scheme provides high flexibility to the evolutionary algorithm. When threshold takes values close to 100% the scheme actually serves as a classic roulette scheme. For values of T close to 10%, the scheme serves close to a classic truncation model. The increased selective pressure that is produced with lower values of T focuses the search on the top individuals, but the genetic diversity is lost and the procedure is trapped in local optima. Such observations and several trial-and-error runs led to the selection of values of T between 40% and 70% in this work.

A. Solid Boundary Representation

The solid terrain under the UAV is most generally represented by a meshed 3-D surface, produced using mathematical functions of the form

$$\begin{aligned} z(x, y) &= \sin(y + a) + b \cdot \sin(x) + c \cdot \cos\Big(d \cdot \sqrt{y^2 + x^2}\Big) \\ &\quad + e \cdot \cos(y) + f \cdot \sin\Big(f \cdot \sqrt{y^2 + x^2}\Big) + g \cdot \cos(y) \end{aligned} \tag{2}$$

where a, b, c, d, e, f, g are constants experimentally defined, in order to produce a surface simulating a rough terrain with mountains and valleys (as shown in Fig. 1).

In order to generate concave terrains (as shown in Fig. 2), the following mathematical function has been used:

$$z(x,y) = \begin{cases} \begin{pmatrix} a \cdot \cos\Big(b\sqrt{x^2+y^2}\Big) \\ -c \cdot \exp(-|y|) - f\exp(-|x-2|) \end{pmatrix} \\ \quad \text{if } |y| < 1.5 \text{ and } 5 < x < 5.5 \\ \begin{pmatrix} a\cos\Big(b\sqrt{x^2+y^2}\Big) \\ +d\sin\Big(g\sqrt{x^2+y^2}\Big) \end{pmatrix} \\ \quad \text{if } 5 < |x| < 8 \text{ or } 5 < |y| < 8 \\ \begin{pmatrix} a \cdot \cos\Big(b\sqrt{(x^2+y^2}\Big) \\ +e|x|^{|\cos(m(x+y))|} - h|x|^{|\sin(n(x+y))|} \end{pmatrix} \\ \quad \text{if } |y| > 4.5 \text{ or } |x| > 4.5 \end{cases} \tag{3}$$

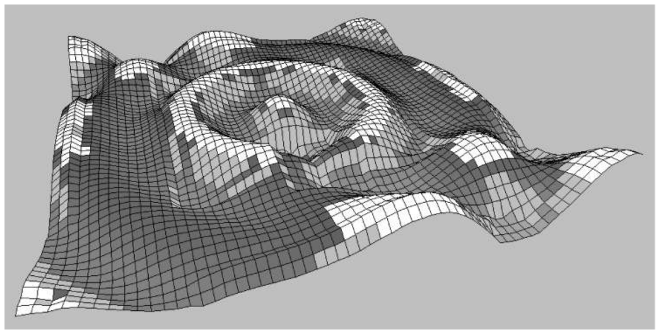

Fig. 2. Artificial terrain produced using (3).

where a, b, c, d, e, f, g, h, m, n are again proper constants, experimentally defined.

A graphical interface has been developed for the visualization of the terrain surface, along with the path line curve as in [9]. The corresponding interface deals with different terrains, produced either artificially or based on real geographical data, and provides an easy verification of the feasibility and quality of each solution. Horizontal sections of the surface in different heights may be plotted, visualizing the boundaries in the UAV flight height (as presented in Fig. 6). The path-planning algorithm considers the scanned surface as a group of quadratic mesh nodes with known coordinates. The same solid boundary representation is used for both the online and the offline method.

B. Path Line B-Spline Modeling

Straight-line segments cannot represent a flying object path line, as usually the case with mobile robots, sea, and undersea vessels. B-Splines are adopted to define the UAV desired path, providing at least first order derivative continuity. B-Spline curves are well fitted in the evolutionary procedure, as they need a few variables (coordinates of the control points) in order to define complicated curved paths (see Appendix B). Each control point has a very local effect on the curve's shape, and small perturbations in its position produce changes in the curve only in the neighborhood of the changing control point [11], [12].

A valuable characteristic of the adopted B-Spline curves is that the curve is tangential to the control polygon at the starting and ending points. This characteristic can be used in order to define the starting direction of the curve, by inserting an extra fixed point after the starting one. These two points can define the direction of the curve at the corresponding region. This is essential for the path planning of flying vessels, as their flight angles are continuously defined. Consequently the direction of the designed path line in the starting position must coincide with the current direction of flight in this position, in order to ensure curvature continuity of the whole path line. The B-Spline curve is discretized, using a constant step, and it is used in this form for the calculation of its fitness.

III. Offline Path Planning

The offline path planner generates a B-Spline curve over known, simulated or real, environments. The starting and ending points of the curve are fixed. A third point close to the starting one is also fixed, determining the initial flight direction. Between the fixed control points, free-to-move control points determine the shape of the curve, taking values in the constrained space. The number of the free-to-move control points is fixed (user-defined). Their physical coordinates are the genes of the EA artificial chromosome, resulting in a fixed length chromosome.

A. Fitness Function

The optimization problem to be solved minimizes a set of four terms, connected to various constraints. The constraints are associated with the feasibility and the length of the path line, a safety distance from the obstacles and the UAVs flight envelope restrictions. The fitness function is the inverse of the weighted sum of the four different terms

$$f = 1 \Big/ \sum_{i=1}^{4} a_i f_i \tag{4}$$

where a_i are weights and f_i are the above mentioned terms defined as follows:

Term f_1 penalizes the nonfeasible curves that pass through the solid boundary. The penalty value is proportional to the number of discretized curve points (not the B-Spline control points) located inside the solid boundary. In this way nonfeasible curves with fewer points inside the solid boundary show better fitness than curves with more points inside the solid boundary. Additionally, the fitter of the nonfeasible curves may survive the selection procedure and produce acceptable offsprings through the heuristic crossover operation.

Term f_2 is the length of the curve (nondimensional with the distance between the starting and destination points) used to provide shorter paths.

Term f_3 is designed to provide flight paths with a safety distance from solid boundaries, given as

$$f_3 = \sum_{i=1}^{nline} \sum_{j=1}^{nground} 1/(r_{i,j}/r_{safe})^2 \tag{5}$$

where $nline$ is the number of discrete curve points, $nground$ is the number of discrete mesh points of the solid boundary, $r_{i,j}$ is the distance between the corresponding nodes and curve points, while r_{safe} is the minimum safety distance from the solid boundary.

Term f_4 is designed to provide curves with a prescribed minimum curvature radius. This characteristic is essential for a flying vessel, as its flight envelope determines the minimum radius of curvature. The angle θ (Fig. 3) that is determined by two successive discrete segments of the curve (defined by the dots in Fig. 3) is calculated and if less than a prescribed value, a penalty is added to the fourth term of the fitness function.

Weights a_i are experimentally determined, using as criterion the almost uniform effect of the last three terms in the fitness value. Term $a_1 f_1$ has a dominant role in (4) providing feasible curves in few generations, since path feasibility is the main concern.

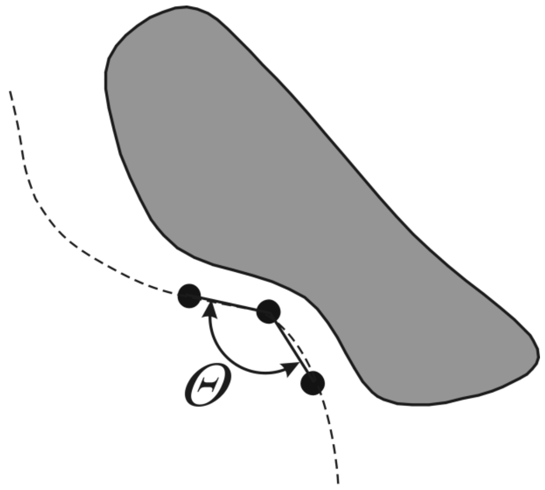

Fig. 3. Schematic representation of the curvature angle used for the calculation of term f_4.

The maximization of Eq. (4), through the EA procedure, results in a set of B-Spline control points, which actually represent the desired path.

B. Offline Procedure

Initially, the starting and ending points are externally determined, along with the direction of flight. The limits of the physical space, where the vehicle is allowed to fly (X, Y, Z, upper and lower coordinates), are also determined, along with the ground surface (in (2) or (3), or real GIS data). The given flight direction is used to determine the third fixed point close to the starting one. Its position is along the flight direction and at a pre-fixed distance from the starting point.

The EA randomly produces a number of chromosomes, equal to the (fixed) population number. The genes of each chromosome are the physical coordinates of the free-to-move B-Spline control points. These coordinates take (random) values within the limits of the constrained physical 3-D space. Each B-Spline curve is constructed by the three fixed control points and the free-to-move ones (provided by the corresponding chromosome).

Each B-Spline is evaluated, using the aforementioned criteria, and its fitness function is calculated. The EA proceeds, as described in Section II.

IV. Online Path Planning

The EA used by the online path planner is based on the one used by the offline path planner. The main modifications concern the representation of the individuals, the initial population, the optimization criteria, and the gradual segment generation of the complete path line. The terrain is considered unknown, and only an area near the UAV is supposed to be scanned by the on board sensors.

A. Representation of the Individual

As the terrain is completely unknown and the radar gradually scans the area, it is impossible to generate a feasible path that connects the starting point with the ending one. Instead, at certain moments, the radar scans a region around the UAV, and a path line is generated that connects a temporary starting point with a temporary ending point. Each temporary ending point is also the next curve starting point. Therefore, what is finally generated is a group of smooth curve segments, connected to each other and eventually, connecting the starting point with the final destination.

In the online problem, only four control points define each B-Spline curve, the first two of which are fixed, determining the direction of the current UAV path. The remaining two control points are allowed to take any position within the scanned by the radar known space, taking into consideration given restrictions. Only the Cartesian coordinates of the nonfixed control points form each individual's genes.

When the next path segment is to be generated, only the first control point of the B-Spline curve is known, as it is the same with the last control point of the previous B-Spline segment. The second control point is not random, as it is used to make sure that at least first derivative continuity of two connected curves is provided at the point that the two curves are connected. Hence, the second control point of the next curve should lie on the line defined by the last two control points of the previous curve. It is also desirable that the second control point is near the first one, so that the UAV may easily avoid any obstacle suddenly sensed in front of it. This may happen because the radar scans the environment not continuously, but at intervals.

B. Initial Population

A random number generator is used to produce floating-point values within certain boundaries. Lower and upper boundaries form the ground area the radar scans within which the UAV is allowed to move. The control points that are randomly generated are acceptable only if they are within the radar's range distance from the UAV. Otherwise, they are ignored and new genes are being generated to replace them. Thus, all individuals represent curves whose control points are within the UAVs radar range.

The initial population size is now smaller compared to the offline path planner one, since the online problem calls for shorter computation times.

C. Online Mechanism

As previously mentioned, the path-planning algorithm considers the scanned surface as a group of quadratic mesh nodes. All ground nodes are initially considered unknown. An algorithm is used to distinguish between nodes visible by the radar and nodes not visible as follows: A node is not visible by the UAV if it is not within the radar's range or if it is within the radar's range but is hidden by a ground section that lies between it and the UAV. The corresponding algorithm, simulates the radar and checks whether the ground nodes within the radar range are "visible" or not and consequently "known" or not.

The radar's data are used to produce the first path line segment. As the UAV is moving along this segment and until it has traveled about 2/3 of its length, the radar scans the surrounding area, returning a new set of visible nodes. The online planner, then, produces a new segment, whose first point is the last point of the previous one and whose last point lies somewhere in the

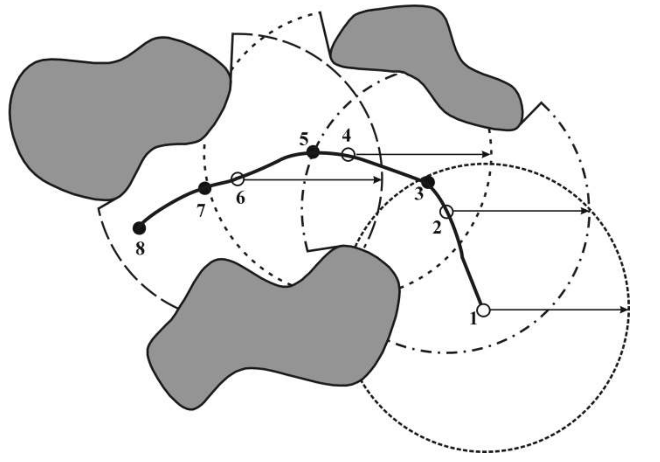

Fig. 4. Schematic representation of the online procedure. The curved solid line is the generated path line, formed by 4 successive B-Spline segments (1–3, 3–5, 5–7, 7–8). Point 1 is the starting position, while point 8 is the ending one. The dashed circles define the radar-covered area in the position where the EA procedure starts to calculate the next B-Spline segment. The corresponding circle centers are marked with white dots (points 1, 2, 4, 6). The ending point of each generated B-Spline segment (points 3, 5, 7) lie close to the maximum radius of each circle (except the final one—point 8).

newly scanned area, its position being determined by the EA online procedure. The process is repeated until the ending point of the current path line segment lies close to the final destination (Fig. 4). The position at which the algorithm starts to generate the next path line segment (here taken as the 2/3 of the segment length) depends on the radar range, the UAVs velocity and the algorithm computational demands.

D. Fitness Function

In this case, the fitness function to be maximized through the EA is the inverse of the weighted sum of eight different terms

$$f = 1 \Big/ \sum_{i=1}^{8} a_i f_i \tag{6}$$

where a_i are the weights and f_i are the corresponding terms described below.

Term f_1 is the same as term f_1 used by the offline path planner.

The value of term f_2 depends on the value of a potential between the starting point and the final target. The potential field between the two points is the main driving force for the gradual development of the path line in the online procedure. The potential is similar to the one between a source and a sink, given as

$$\Phi = \ln \frac{r_2 + c \cdot r_0}{r_1 + c \cdot r_0} \tag{7}$$

where r_1 is the distance between the last point of the current curve and the starting point, r_2 is the distance between the last point of the current curve and the final destination, r_0 is the distance between the starting and final destination and c is a constant.

This potential allows for selecting curved paths that bypass obstacles lying between the starting and ending point of each

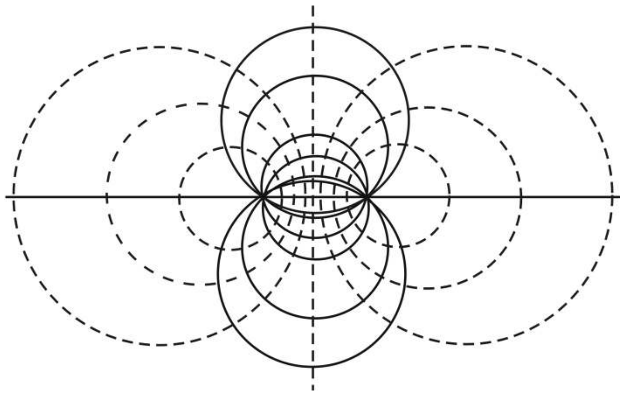

Fig. 5. Potential field of (7). The dashed lines are the equal-potential lines, while the solid lines are the "streamlines" normal to the previous ones. The points in the intersection of the streamlines are the starting and the final ones. All the points that lie on an equal-potential curve have the same probability for being chosen as a final point for each B-Spline segment (according to term f_2 of the online procedure).

B-Spline curve. A visualization of the corresponding potential field is demonstrated in Fig. 5.

Term f_3 is the same as term f_2 used by the offline path planner and is applied to each segment of the total path line. Term f_4 is the same as term f_3 used by the offline path planner. Term f_5 is a function of the distance between the current path's last point and the solid boundary, similar to term f_4, and it penalizes a curve that ends too close to it. The purpose of this term is to help the UAV avoid moving too close to the ground surface. Hence, while the UAV traverses its next partial trajectory, it does not change its direction abruptly in order to avoid a collision with the ground.

Term f_6 is designed to prevent the UAV from being trapped in local optima and to force it move toward unexplored areas. It may be possible that some segments of the path line are concentrated in a small area, away from the final target. In order to help the UAV leave this area, term f_6 repels it from the points of the path line it has already traversed. Furthermore, if the UAV is wandering around to find a path that will guide it to its target, the UAV will be forced to move toward areas it has not visited before. This term has the form

$$f_6 = \sum_{k=1}^{npoint} \frac{1}{r_k} \tag{8}$$

where $npoint$ is the number of the discrete curve points produced so far and r_k is their distance from the last point of the current curve segment.

Term f_7 is the same as term f_4 used by the offline path planner. If the calculated angle is less than a prescribed value (depending on flight restrictions of the considered UAV), a penalty is added to the seventh term of the fitness function. The flight envelope of the UAV determines the minimum acceptable angle.

Term f_8 represents another potential field, which is developed in a small area around the final target. When the UAV is away from the final target, the term is given a constant value. When the UAV is very close to the target, the term's value decreases proportionally to the square of the distance between the current curve's last point and the target. Thus, when the UAV is near its target, the value of this term is quite small and prevents the UAV from moving away.

Weights a_i are experimentally determined, using as criterion the almost uniform effect of all the terms, except the first one. Term $a_1 f_1$ has a dominant role, in order to provide feasible curve segments in a few generations, since path feasibility is the main concern.

E. Local Optima Avoidance Mode of the Online Path Planner

Although term f_6 has been introduced in the fitness function of the online procedure, the ground formation may cause the UAV to be trapped in a local optimum area (consider a mountain with a horseshoe shape as in Fig. 2 and that the UAV enters inside its concave part, while it is not possible to overpass the mountain). In such cases, it is difficult for the UAV to escape from this area as it would have to move away from the obstacle and thus away from its final target. To prevent this from happening, a second fitness function and an algorithm is derived to predict when the UAV is about to be trapped in a local optimum and use a second mode of the online planner to overcome it.

As already stated, every time a new segment is generated, and the radar has scanned its environment, it checks whether there is an "obstacle" very close (within a predefined safety distance) to the UAV and toward its motion direction. An "obstacle" is a group of ground nodes higher than the flight altitude that the UAV cannot overpass, due to the constrained space in which it is allowed to move. As term f_5 has been added to the fitness function, (6), which does not allow the UAV to come too close to the ground. It is reasonable to believe that the UAV will come too close to the ground only if it is trapped in a local optimum area. In addition, it is being checked whether there is an obstacle close to the UAV, and on its way to the final destination. If no obstacle is found, the fitness function, given by (6), guides the UAV.

The goal of the second mode of the algorithm is to force the UAV to move along the border of the obstacle that has caused the UAV to be trapped, until that obstacle lies no longer between the UAV and its final destination. This is achieved by identifying the obstacle that has caused the UAV to be trapped. An algorithm, presented in Appendix A, generates a map of all groups of nodes that form obstacles that are visible by the radar. Each such group of nodes contains nodes that lie adjacent to each other. After this procedure terminates, each group of nodes has been given a unique number. All the nodes of a group have been given the same number. As the coordinates of the node that lies on the UAVs motion direction have been stored, it is known which group of nodes (obstacle) has caused the UAV to be trapped in the local optimum area (see Appendix A).

Next, the borders (the corresponding nodes) of the obstacle are located and assigned a numerical value, so that the node with the greater value is far from the UAVs current position. The second fitness function forces the UAV to move along the border and toward the border node with the larger value.

When the UAV reaches its new position, the whole process goes over again (in the same mode) and the UAV keeps moving across the obstacle's borders, (momentarily) ignoring its final destination. The algorithm returns to the first mode when the obstacle is no longer between the UAV and the final destination.

F. Second Fitness Function of the Online Procedure

The second fitness function is used within the local optima avoidance mode of the online procedure, when the UAV is trapped in a local optimum.

It is the inverse of the weighted sum of three different terms

$$f = 1 \Big/ \sum_{i=1}^{3} a_i' f_i' \qquad (9)$$

where a_i' are the weights and f_i' are the corresponding terms described below. Term f_1' is the same as the term f_1 used by the offline path planner. The value of term f_2' penalizes curves when their last point is far from the border node whose identification number was given the greater value. This term is used to enforce the curve to circle the obstacle.

Term f_3' is the nondimensional length of the current curve, used in order to provide short and smooth paths. Weights a_i' are experimentally determined. Term $a_1' f_1'$ has a dominant role, in order to provide feasible curves in a few generations, since path feasibility is the main concern.

The second fitness function works in a different way than term f_6 of the first fitness function. Term f_6 repels the UAV from the points of the path line it has already traversed, in order to explore new areas, and avoid "going around" in the local optimum area. The second fitness function forces the path line to move along the border of the obstacle that has caused the UAV to be trapped, until that obstacle no longer lies between the UAV and its final destination.

This second fitness function is used to generate a new path line segment if the UAV has been trapped in a local optimum area. After the calculation of the new path line segment, using the second fitness function, the UAV moves along this segment, until it reaches the 2/3 of segment's length. If the obstacle is no longer between the current UAV position and its final destination, the algorithm returns to the first mode of the online procedure, and the fitness function defined in (6) is used. If the obstacle is still between the UAV and the final destination, the next path line segment will be generated with the fitness function defined in (9) (the second one), within the second mode of the online procedure.

V. Experimental Results

A. Offline Experiments

The offline planner has been extensively tested, using a simulation environment. All experiments have been designed in order to search for path lines between "mountains." For this reason, an upper ceiling for flight height has been enforced. This ceiling is represented in the graphical environment by the horizontal section of the terrain (Fig. 6).

The (experimentally optimized) settings of the evolutionary algorithm for the offline planner are as follows: *population size* $= 100$, *threshold* $= 0.5$, *heuristic crossover probability* $= 0.75$, *classic crossover probability* $= 0.25$, *classic mutation probability* $= 0.05$, *nonuniform mutation probability* $= 0.13$. The detailed procedure of the parameter selection is presented in Appendix C.

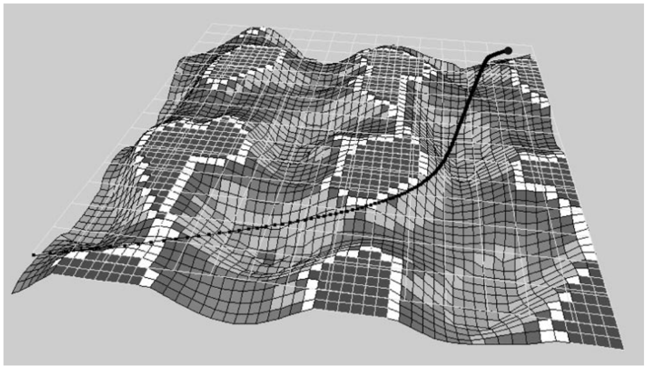

Fig. 6. First offline test case: Four free-to-move control points were used, with a prefixed direction at the starting position. The starting position is marked with a circle.

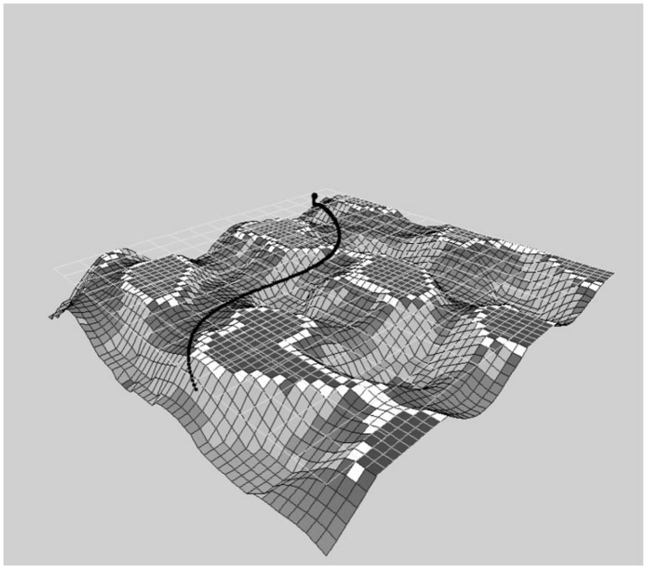

Fig. 7. Second offline test case.

The algorithm was defined to terminate after 50 generations, although feasible solutions can be reached in less than 10 iterations. With 50 generations, and a population size equal to 100, 5000 evaluations of the fitness function are performed before the algorithm stops. The calculation corresponds to 15 s computation time per generation, in a 700 MHz PC, for a chromosome length equal to 12 (4 free-to-move control points), and a terrain described with 53 × 53 nodes.

In the offline test cases presented here, the free-to-move control points were taking values between 4–6, resulting in a total number of B-Spline control points equal to 7–9 (along with the fixed starting and target points, and the fixed second point, used for the determination of the initial direction). Greater number of control points resulted in higher computation time and slower convergence rate, without any significant profit, concerning the fitness of the curve.

The test cases shown in Figs. 10 and 11 were designed with the ceiling set to a low altitude which increased the path planning difficulty. The minimum distance from the mountain-like boundaries was empirically set equal to 1/30 of the x-dimension of the terrain, in all the cases, except for case 3, which is presented in Figs. 8 and 9. In test case 3, the minimum distance

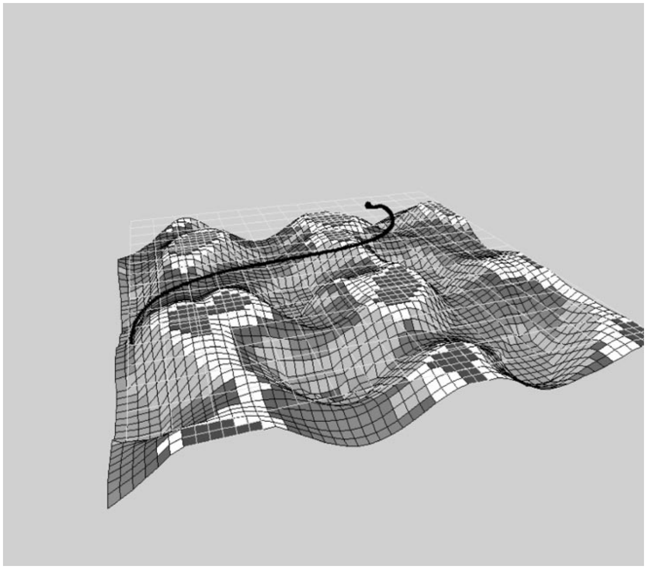

Fig. 8. Third offline test case.

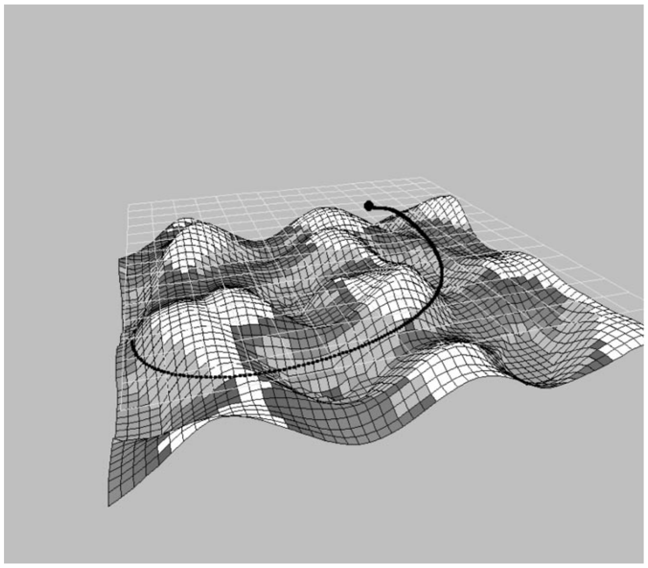

Fig. 9. Early feasible solution of the third offline test case.

was set equal to the 1/15 of the terrain's x-dimension. As it is demonstrated in Fig. 8, a higher distance from the solid boundaries was achieved, compared to test case 2 (Fig. 7). For the test case 5, shown in Fig. 11, a wider terrain was used. For the test case 6, shown in Fig. 12, a horseshoe terrain was used [produced with (3)], demonstrating the ability of the method to deal with U-shaped obstacles.

Relatively high values of mutation probabilities were adopted, in order to ensure the ability of the algorithm to overcome local optima. As it was observed, initial feasible solutions provided by the evolutionary algorithm, were progressively replaced by fitter ones, with a completely different structure. Figs. 8 and 9 demonstrate the above observation.

B. Online Experiments

The same environment was used for all the test cases considered, with different starting and destination points and different

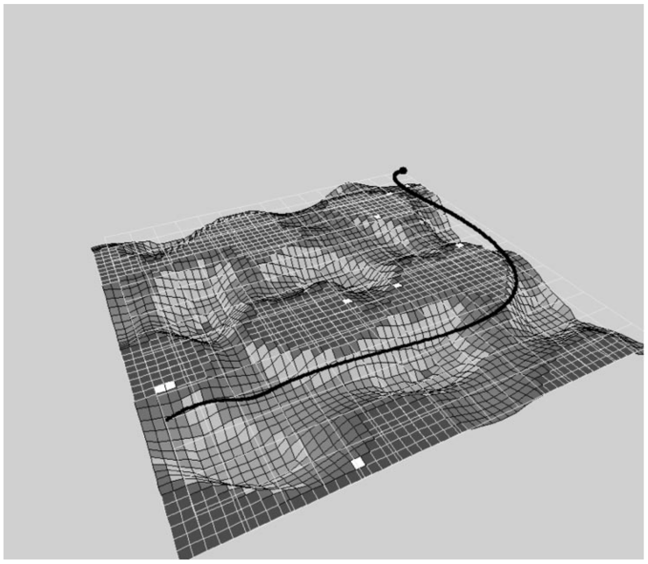
Fig. 10. Fourth offline test case, with 4 free-to-move control points, and a very low upper limit.

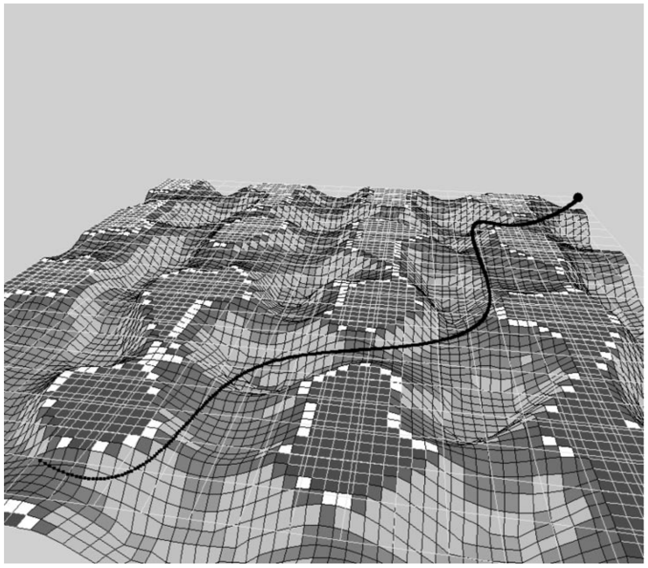
Fig. 11. Wider terrain was used for the fifth offline test case, along with 6 free-to-move control points and a very low upper limit.

initial flight directions. All the test cases presented here were designed with a maximum flight altitude, as in the offline procedure (different for each case).

The *population size* was set equal to 50, while the algorithm was defined to terminate after 25 generations, although feasible solutions can be reached in less than ten generations for each curve. The lower values for the population size (50) and for the maximum number of generations (25), compared to the offline procedure (100 and 50 respectively), were adopted in order to minimize the computational time, which is essential for online applications. The shorter length of the chromosomes (only six genes—two control points) and the narrow search space (defined by the radar range), compared to the offline procedure, made possible the aforementioned reduction. For the calculation of each segment of the path line (which needs a complete EA

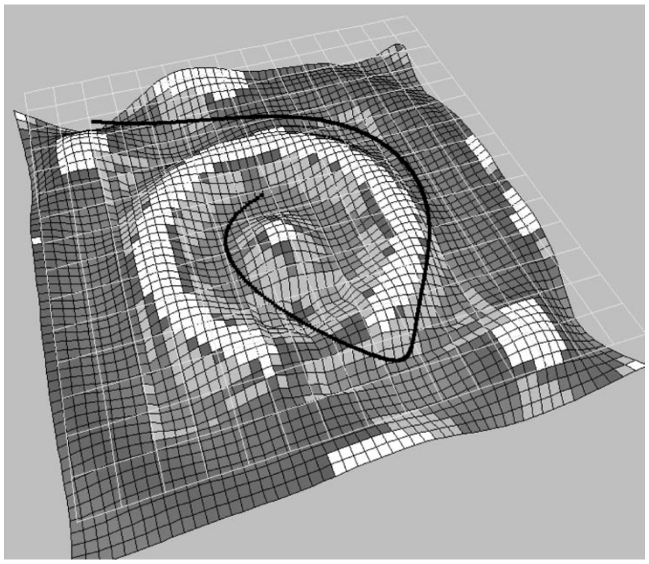
Fig. 12. Different terrain was used for the sixth offline test case, demonstrating the ability of the method in dealing with difficult obstacle shapes.

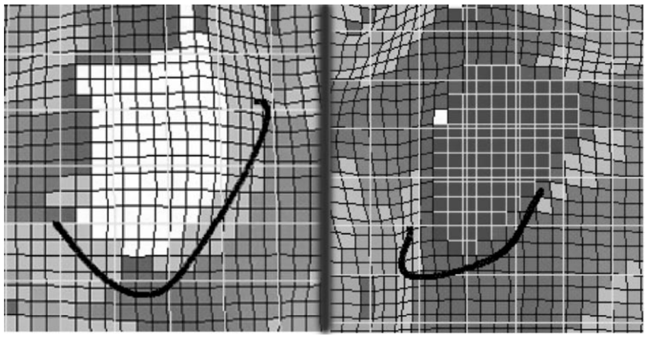
Fig. 13. Preliminary test cases for the second mode of the online procedure. The figure demonstrates the ability of the second mode of the online procedure in driving the UAV to by-pass an obstacle. The starting positions are on the left side of each obstacle.

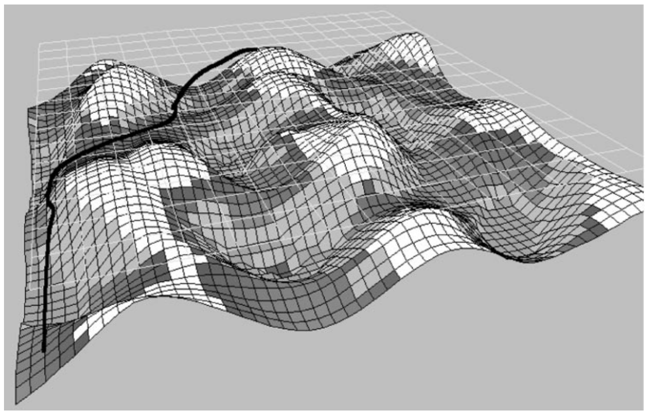
Fig. 14. First test case for the online path planner: Starting position is near the left corner.

computation), less than five seconds per generation are needed, in a 700 MHz PC.

In order to check the validity of the second mode of the online procedure, specialized test cases were used. In these test cases the starting and ending points of the path line were positioned in the opposite side of a “mountain,” while an upper flight

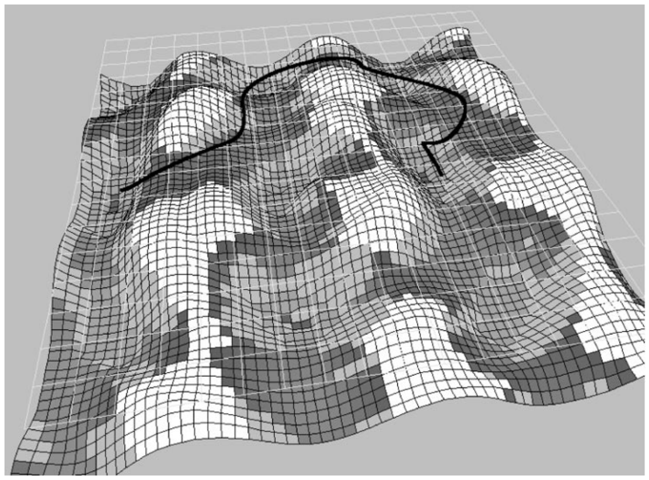
Fig. 15. Second test case for the online path planner: Starting position is near the left side.

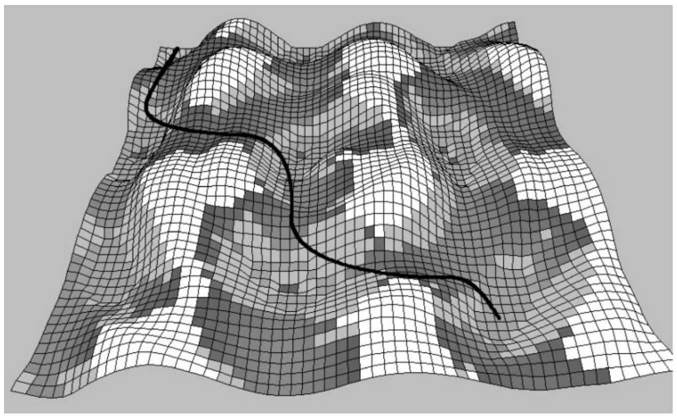
Fig. 16. Third test case for the online path planner: Starting position is near the upper corner.

ceiling was enforced. The results for two of the aforementioned test cases are shown in Fig. 13. As demonstrated, the procedure produces a smooth path (consisted of 2–4 successive B-Spline curve segments), bypassing the solid obstacle and reaching effectively the final target. It is obvious that the UAVs radar cannot "see" the final target from the starting position, and the knowledge of the obstacle's borders is gained gradually.

For the selection and optimization of the various terms (constants and weights used in the first fitness function of the online procedure), two initial test cases were adopted with their results shown in Figs. 14 and 15. In test cases 3–5 (Figs. 16–18), only the first mode of the online procedure was used. The path lines consist of more than 6 B-Spline curve segments, which are smoothly connected to each other. In the last test case considered (Fig. 19), the algorithm passed to the second mode, in order to avoid an obstacle (at the second abrupt turn). As it can be observed, the UAV surrounds the obstacle and then the first mode of the algorithm takes over, in order to guide it to the final destination. The abrupt turns of the path line are due to the fact that the last point of the corresponding B-Spline curve segments is very close to the terrain border (the first) and the obstacle (the second), while a solution to this malfunction is under consideration.

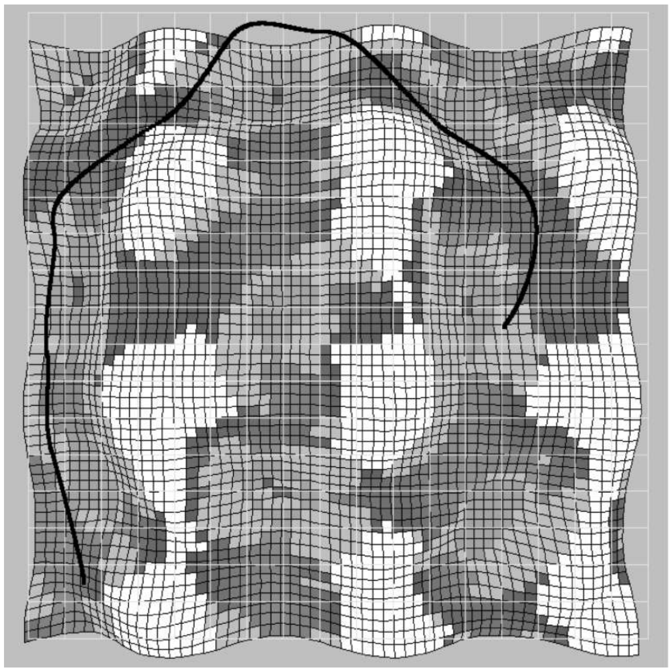
Fig. 17. Fourth test case for the online path planner: Starting position is near the left corner.

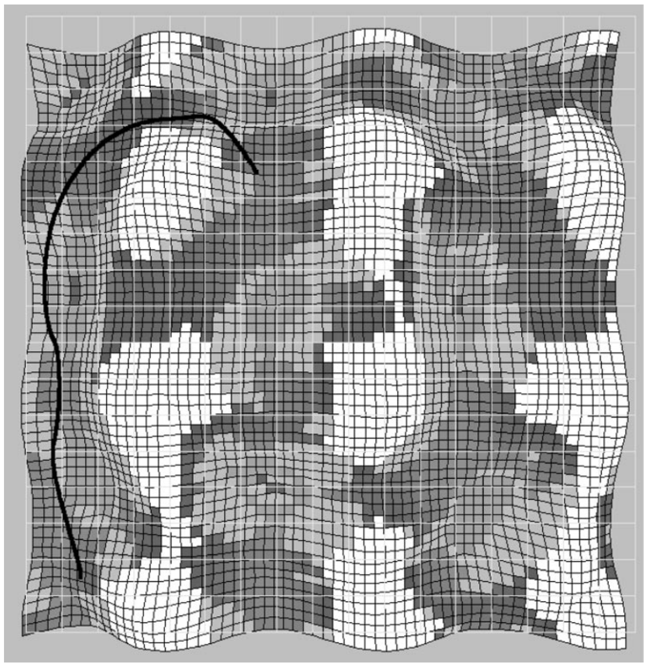
Fig. 18. Fifth test case for the online path planner: Starting position is near the left corner.

VI. Discussion

An evolutionary algorithm-based offline/online path planner for unmanned aerial vehicles (UAVs) has been presented to calculate a curved path line with desired characteristics in a 3-D rough terrain environment.

The trajectory of a UAV cannot be, adequately, represented using line segments. Additionally, a flying vessel cannot follow a path line formed with straight line segments, without giving

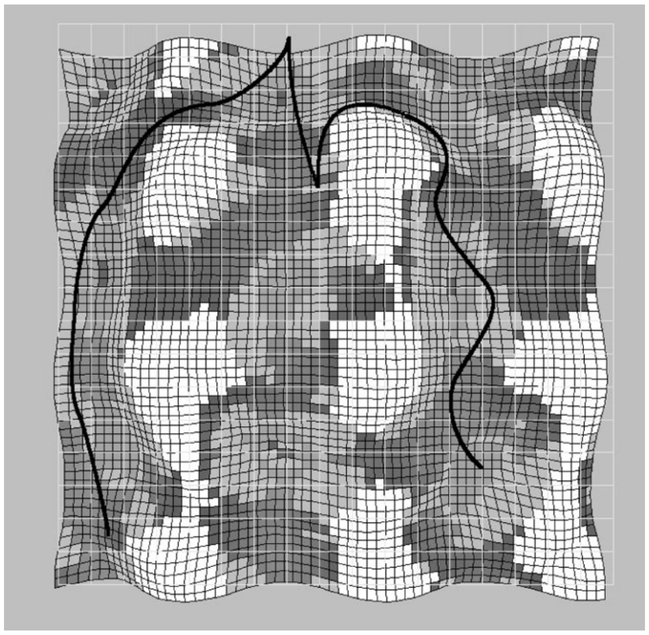
Fig. 19. Sixth test case for the online path planner: Starting position is near the left corner.

rise to control and stability problems. The proposed method uses parametric curves to produce a continuous path line, which is described by a small set of parameters—the coordinates of the control points. The construction of the B-Splines based on control points, proved suitable for coupling with an EA. The direction of the curve can easily be prescribed at its starting position, by inserting a second fixed point. The direction described by these points is the initial direction of each curve and must coincide with the current flight direction.

The offline planner takes into consideration the vehicle flight capabilities in the form of a prescribed minimum curvature angle and a maximum flight height. The resulting path is smooth and assumed to be easily generated by autonomous navigation controllers. The online path planner, although it evaluates a minimum curvature angle, associated with the flight envelope, may produce paths that exceed the vessel's capabilities. In real-life implementations, the tuning of the optimization procedure is heavily based on UAVs speed and maneuverability as much as sensor's range and accuracy. Reactive-based control methods may be useful during the collision avoidance procedure. Such methods are proved to cooperate well with path planners when applied to ground robots [16].

The EA proved to be effective in finding feasible path lines (for both offline and online procedures) under the forced constraints and within an acceptable time period, especially for the online planner, where the execution time is of great importance. The easy implementation of the various constraints of the problem proved to be a valuable characteristic of EA. Nevertheless, a feasible solution could be reached within a small number of iterations, while the rest of the iterations were used in order to optimize the solution, according to the rest of the criteria.

The introduced potential field between the initial and target position was the main driving force for the gradual development of the path line in the online procedure. As it was demonstrated by the several test cases presented, it proved to be effective in producing curves that bypass the solid ground obstacles positioned between the starting and target positions, provided that they do not have a concave shape, and do not drive the fitness function in local optima. In this last case the second mode of the online procedure governs the path planning, producing smooth flight paths (consisting of successive B-Spline curves) that by-pass the obstacle.

The proposed method proved capable for producing feasible (collision-free) paths after a small number of generations (less than 10), rendering it suitable for real-time calculations. The latter is particularly useful in online applications, where the global terrain geography is unknown and a local knowledge is gained through onboard sensors.

Appendix A

The algorithms used for obstacle identification and enumeration of border nodes are presented.

If there is a node-obstacle close to the UAV and along its motion direction, its coordinates are stored. Then, the following algorithm is used to generate a map of all groups of nodes that form obstacles and are visible by the radar. Each group (obstacle) contains nodes that lie adjacent to each other.

```
Do the following until all segments $(i, j)$ that are visible by the UAVs radar have been checked, starting from the upper-right segment of the visible terrain.
If node $S$ is an obstacle then
        If it is the first obstacle checked, then the variable groupvalue$(i, j) = 1$ and the variable $\max = 1$.
Else check whether there is another node-obstacle either on its right or up-right or upper or up-left position.
          If there is, then read its groupvalue$(i, j)$ value. Set the current node's $(S)$ groupvalue$(i, j)$ value equal to the one of the neighboring obstacle that was found in the previous step.
          Else set groupvalue$(i, j) = \max + 1$, where $\max$ is the greater value that the variable groupvalue$(i, j)$ has been given so far.
      End-If
End-If
Loop
      Do until all the nodes $(i, j)$ that are visible by the radar have been checked, starting from the lower-down node of the terrain.
   If node $S$ is an obstacle then
        Check whether there is another node-obstacle either on its right or upper or up-left or up-right position.
        If there is, then set the node's groupvalue$(i, j)$ value equal to the one of segment $S$.
   End-If
Loop
```

After this process has finished, each group of nodes (obstacles) has been given a unique number [equal to the *groupvalue*(i, j) value of the nodes that are part of it]. All the nodes of a group have been given the same number. As the coordinates of the node that lies on the UAVs motion direction have been kept, it is known which group of nodes (obstacle) has caused the UAV to be trapped in a local optimum area.

Next, the obstacle's borders are located and are enumerated so that the node with the greater number is far from the UAVs current position. The following algorithm is used in order to enumerate the nodes of the obstacle that are part of its border.

A. Counter-Clockwise Numeration

Give the value 0 to the variable num for the obstacle that lies on the UAVs motion direction and whose coordinates have been already stored.
Check whether there is another border-node S of the group of obstacles in the 8 surrounding positions, using counter clockwise direction and starting from the lower position.
If there is, then the variable num is given the value $num = num + 1$ and the same process is repeated for the new node S. The process is repeated until there is no other border-node adjacent to the current node S for which the variable num has not been given a value.

If the current node S lies on the border of the terrain, then clockwise numeration is needed, so as the UAV not to move outside the terrain.

B. Clockwise Numeration

Give the value 0 to the variable num for the obstacle that lies on the UAVs motion direction and whose coordinates have been already kept.
Check whether there is another segment-border S of the group of obstacles, in the 8 surrounding positions, using clockwise direction and starting from the upper position.
If there is, then the variable num is given the value $num = num + 1$ and the same process is repeated for the new segment S. The process is repeated until there is no other segment-border adjacent to the current segment S for which the variable num has not been given a value.

Appendix B

B-Spline curves are parametric curves, with their construction based on blending functions [11], [12]. Their parametric construction provides the ability to produce nonmonotonic curves. If the number of control points of the corresponding curve is $(n + 1)$, with coordinates $(x_0, y_0, z_0), \ldots, (x_n, y_n, z_n)$, the coordinates of the B-Spline may be written as

$$X(t) = \sum_{i=0}^{n} x_i \cdot B_{i,K}(t) \quad (10)$$

$$Y(t) = \sum_{i=0}^{n} y_i \cdot B_{i,K}(t) \quad (11)$$

$$Z(t) = \sum_{i=0}^{n} z_i \cdot B_{i,K}(t) \quad (12)$$

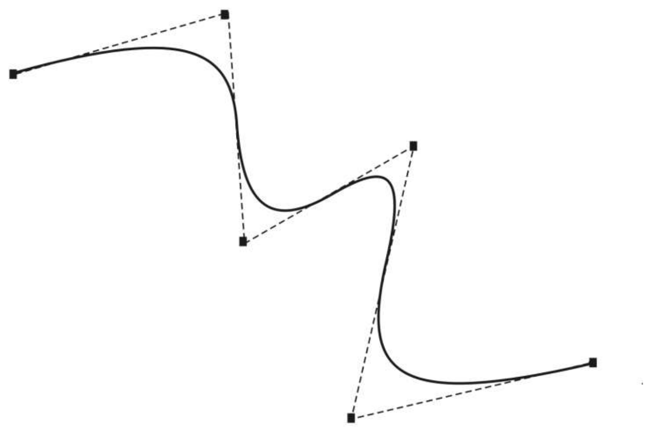
Fig. 20. 2-D quadratic B-Spline curve ($K = 3$), with its control points and the control polygon.

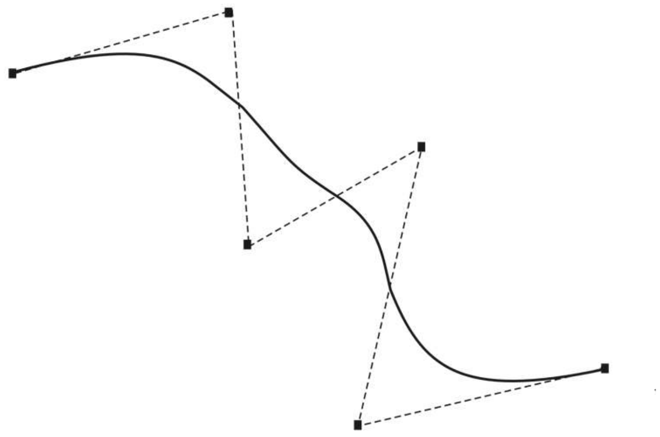
Fig. 21. Corresponding B-Spline curve with the same control points as in Fig. 20 for $K = 5$.

where $B_{i,K}(t)$ the blending functions of the curve and K the order of the curve, which is associated with curve's smoothness. Higher values of K correspond to smoother curves, as it is demonstrated in Figs. 20 and 21. Parameter t varies between 0 and $(n - K + 2)$ with a constant step, providing the discrete points of the B-Spline curve. The sum of the values of the blending functions for any value of t is always 1.

The blending functions are defined recursively in terms of a set of *Knot* values, with the most common form being the *uniform nonperiodic* one, defined as:

$$\begin{aligned} Knot(i) &= 0, && \text{if } i < K \\ Knot(i) &= i - K + 1, && \text{if } K \le i \le n \\ Knot(i) &= n - K + 2, && \text{if } n < i. \end{aligned} \quad (13)$$

The blending functions $B_{i,K}$ are defined recursively, using the *Knot* values given by (13)

$$B_{i,1}(t) = \begin{cases} 1, & \text{if } Knot(i) \le t < Knot(i+1) \\ 1, & \text{if } \begin{cases} Knot(i) \le t \le Knot(i+1) \\ \text{and} \\ t = n - K + 2 \end{cases} \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

TABLE I
EPERIMENTAL RESULTS FOR REURISTIC CROSSOVER PROBABILITY

heuristic crossover probability	fitness function at 30 generations	number of generations for feasible solution
0.25	0.755846	22
0.50	0.038317	>30
0.70	0.890083	16
0.75	**0.891374**	**9**
0.80	0.010973	>30
0.85	0.010984	>30

population number = 100
number of generations = 30
non-uniform mutation probability = 0.15
classic crossover probability = 0.15
classic mutation probability = 0.05

TABLE II
EPERIMENTAL RESULTS FOR NON-UNIFORM MUTATION PROBABILITY

non-uniform mutation probability	fitness function at 30 generations	number of generations for feasible solution
0.10	0.010954	>30
0.13	**0.980585**	**11**
0.15	0.891373	9
0.17	0.863955	14
0.20	0.016226	>30

population number = 100
number of generations = 30
heuristic crossover probability = 0.75
classic crossover probability = 0.15
classic mutation probability = 0.05

TABLE III
EPERIMENTAL RESULTS FOR CLASSIC CROSSOVER PROBABILITY

classic crossover probability	fitness function at 30 generations	number of generations for feasible solution
0.05	0.91033	8
0.10	0.967696	10
0.15	0.980585	11
0.20	0.91109	11
0.25	**0.984536**	**7**
0.50	0.047208	>30

population number = 100
number of generations = 30
heuristic crossover probability = 0.75
non-uniform mutation probability = 0.13
classic mutation probability = 0.05

TABLE IV
EPERIMENTAL RESULTS FOR CLASSIC MUTATION PROBABILITY

classic mutation probability	fitness function at 30 generations	number of generations for feasible solution
0.01	0.96730	11
0.03	0.983569	8
0.05	**0.984536**	**7**
0.07	0.902685	9
0.10	0.95480	26
0.20	0.84433	15

population number = 100
number of generations = 30
heuristic crossover probability = 0.75
non-uniform mutation probability = 0.13
classic crossover probability = 0.25

$$B_{i,K}(t) = \frac{(t - Knot(i)) \times B_{i,K-1}(t)}{Knot(i+K-1) - Knot(i)} + \frac{(Knot(i+K) - t) \times B_{i+1,K-1}(t)}{Knot(i+K) - Knot(i+1)}. \quad (15)$$

If the denominator of either of the fractions is zero, that fraction is defined to have zero value.

APPENDIX C

In order to optimize the probabilities of the various EA operators, a simplified experimental procedure was adopted (using a reference offline test case), which is described below. The optimization criteria were the value of the fitness function after 30 generations and the first generation at which a feasible solution was reached. The population number was set equal to 100 (without optimization).

Based on prior experience, from other EA applications, an initial set of parameters was used: *nonuniform mutation probability* $= 0.15$, *classic mutation probability* $= 0.05$, *classic crossover probability* $= 0.15$. The results for different values of *heuristic crossover probability* are presented in Table I. The best results correspond to a *heuristic crossover probability* equal to 0.75, which was fixed for the next calculations.

The procedure was repeated for the *nonuniform mutation probability,* with the results presented in Table II. The best results were obtained for a *nonuniform mutation probability* equal to 0.13, which was fixed for the next calculations.

The procedure was repeated for the *classic crossover probability,* with the results presented in Table III. As it is demonstrated, the best results were obtained for a *classic crossover probability* equal to 0.25.

Finally, as it is demonstrated in Table IV, the corresponding experiments resulted in a *classic mutation probability* equal to 0.05 (with the previous parameters taking values equal to 0.75, 0.13 and 0.25 respectively).

Fig. 22 presents the path line produced using the initial set of probabilities (*heuristic crossover probability* $= 0.75$, *classic crossover probability* $= 0.15$, *nonuniform mutation probability* $= 0.15$, *classic mutation probability* $= 0.05$). Fig. 23 presents the path line produced with the final (optimized) set (for the same test case). The higher fitness function of the optimized set of parameters corresponds to a smoother path line in Fig. 23. The EA convergence histories for the initial and final sets of parameters are presented in Fig. 24. The steps in the curves denote the first generation where a feasible solution was obtained.

Fig. 25 demonstrates the influence of the various EA operators on the convergence rate of the algorithm. The same test case was used as a reference, with a *population number* equal to 100 and a maximum *number of generations* equal to 30. The use of

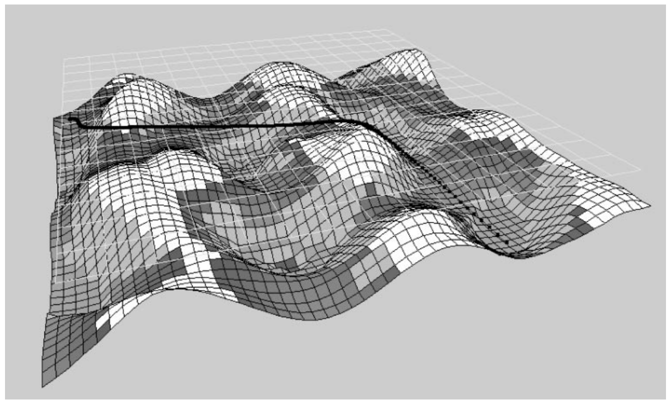

Fig. 22. Path line produced with the initial set of parameters of the EA (heuristic crossover probability = 0.75, nonuniform mutation probability = 0.15, classic crossover probability = 0.15, classic mutation probability = 0.05, generations = 30). The starting position is near the left corner.

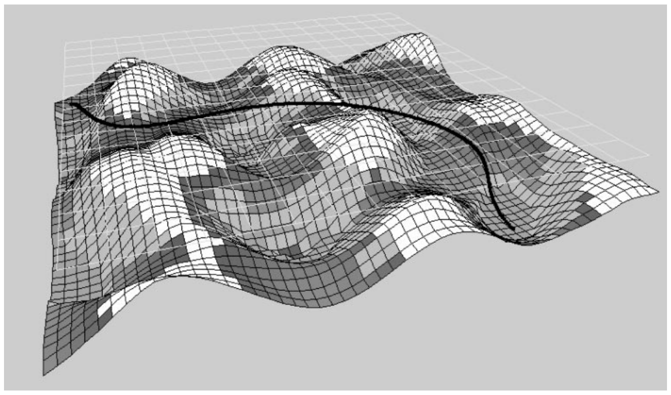

Fig. 23. Path line produced with the final (optimized) set of parameters of the EA (heuristic crossover probability = 0.75, nonuniform mutation probability = 0.13, classic crossover probability = 0.25, classic mutation probability = 0.05, generations = 30).

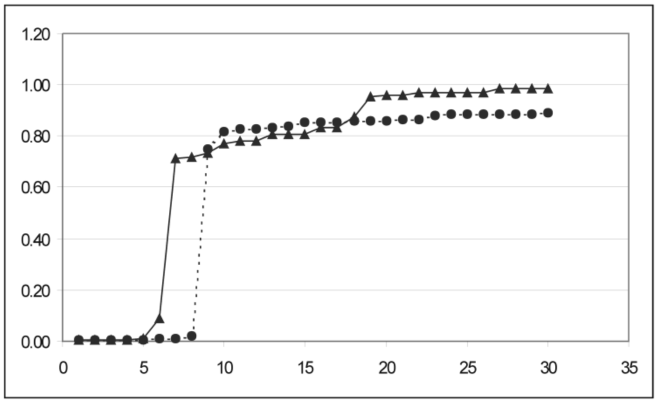

Fig. 24. Convergence histories of the EA, corresponding to the initial (circles) and optimized (triangles) set of parameters.

only the classic mutation and crossover operators proved inadequate to obtain a feasible solution in less than 30 generations. Curve 1 of Fig. 25 shows the convergence rate of such an EA, with a *classic (one-point) crossover probability* equal to 0.8 and a *classic mutation probability* equal to 0.2.

The adoption of the heuristic crossover operator proved to be critical for the enhancement of the convergence rate. Additionally, a feasible solution was obtained in ten generations. Curve 2 of Fig. 25 shows the convergence rate of the corresponding algorithm, with a *heuristic crossover probability* equal to 0.75, a *classic (one-point) crossover probability* equal to 0.25 and a *classic mutation probability* equal to 0.2.

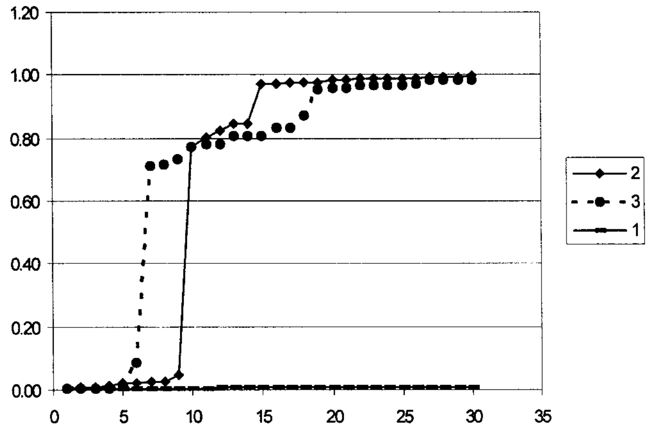

Fig. 25. Influence of the various EA operators in the convergence rate of the algorithm. Curve (1) corresponds to an EA with only classic crossover and mutation operators. Curve (2) corresponds to an EA with an additional heuristic crossover operator, while curve (3) was obtained with the final EA (with all four operators).

The introduction of the nonuniform mutation operator moved the first feasible solution to a lower generation number (generation 7), as it is demonstrated in Fig. 25 (curve 3). The value of the fitness function is practically the same with the previous one of curve 2, after 30 generations. The corresponding EA consists of a heuristic crossover operator with a *probability* equal to 0.75, a classic crossover operator with a *probability* equal to 0.25, a *nonuniform mutation probability* equal to 0.13 and a *classic mutation probability* equal to 0.05.

ACKNOWLEDGMENT

The authors would like to thank the referees for their constructive comments.

REFERENCES

[1] J. Gomez Ortega and E. F. Camacho, “Mobile robot navigation in a partially structured static environment, using neural predictive control,” *Control Eng. Practice*, vol. 4, no. 12, pp. 1669–1679, 1996.

[2] Y. D. Kwon and J. S. Lee, “online evolutionary optimization of fuzzy control system based on decentralized population,” *Intell. Automa. Soft Comput.*, vol. 6, no. 2, pp. 135–146, 2000.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1999.

[4] R. Smierzchalski, “Evolutionary trajectory planning of ships in navigation traffic areas,” *J. Marine Sci. Technol.*, vol. 4, pp. 1–6, 1999.

[5] R. Smierzchalski and Z. Michalewicz, “Modeling of ship trajectory in collision situations by an evolutionary algorithm,” *IEEE Trans. Evol. Comput.*, vol. 4, pp. 227–241, Sept. 2000.

[6] K. Sugihara and J. Smith, “Genetic algorithms for adaptive motion planning of an autonomous mobile robot,” in *Proc. IEEE Int. Symp. on Computational Intelligence Robotics Automation*, Monterey, CA, 1997, pp. 138–143.

[7] K. Sugihara and J. Yuh, “GA-based motion planning for underwater robotic vehicles,” in *UUST-10*, Durham, NH, September 1997.

[8] H. Martinez-Alfaro and S. Gomez-Garcia, “Mobile robot path planning and tracking using simulated annealing and fuzzy logic control,” *Expert Syst.*, vol. 15, pp. 421–429, 1988.

[9] I. K. Nikolos, N. Tsourveloudis, and K. P. Valavanis, "Evolutionary algorithm based 3-D path planner for UAV navigation," in *Proc. CD-ROM 9th Mediterranean Conf. Control Automation*, Dubrovnik, Croatia, 2001.

[10] J. Z. Sasiadek and I. Duleba, "3-D local trajectory planner for UAV," *J. Intelligent Robotic Systems*, vol. 29, pp. 191–210, 2000.

[11] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*. New York: Academic, 1988.

[12] L. Piegl and W. Tiller, *The Nurbs Book*, 2nd ed: Springer, 1997.

[13] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 18–28, Apr. 1997.

[14] A. C. Nearchou, "Adaptive navigation of autonomous vehicles using evolutionary algorithms," *Artif. Intell. in Eng.*, vol. 13, pp. 159–173, 1999.

[15] I. Ashiru, C. Czarnecki, and T. Routen, "Characteristics of a genetic based approach to path planning for mobile robots," *J. Net.Comput. Appl.*, vol. 19, pp. 149–169, 1996.

[16] N. C. Tsourveloudis, K. P. Valavanis, and T. Hebert, "Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 490–497, Aug. 2001.

Ioannis K. Nikolos received the B.S. degree in mechanical engineering from the National Technical University of Athens, NTUA, Athens, Greece in 1990, and the Ph.D. degree in fluid mechanics, Lab. of Thermal Turbomachines, from the Mechanical Engineering Department of NTUA in 1996.

He is an adjunct Assistant Professor with the Department of General Science and the Department of Production Engineering and Management, Technical University of Crete, Chania, Greece. His research interests are in the fields of turbomachines, fluid mechanics, optimization and artificial intelligence.

Dr. Nikolos is a member of the Technical Chamber of Greece and the Greek Society of Mechanical Engineers. His research has been funded by the European Union and the Hellenic GSRT.

Kimon P. Valavanis (SM'80–M'83–SM'91) received the B.S. degree in electrical engineering from the National Technical University of Athens, Athens, Greece in 1981, the M.Sc. and Ph.D. degrees in electrical engineering and computer and systems engineering, from Rensselaer Polytechnic Institute, Troy, NY, in 1984 and 1986, respectively.

Currently, he is a professor at the Department of Production Engineering and Management, and Director of the Intelligent Systems and Robotics Laboratory, Technical University of Crete, Greece. He is also an Adjunct Professor at the Center for Advanced Computer Studies, University of Louisiana at Lafayette and Guest Professor of the Dept. of Telecommunications, Faculty of EE and Computing, University of Zagreb, Croatia. His research interests are in the areas of robotics, automation, and distributed intelligence systems. He has published more than 200 technical papers, book chapters, and technical reports. He has been the general chair and the program chair of IEEE conferences and symposia, and an editor of several conference proceedings. He is the 2003 Mediterranean Conference on Control and Automation General Co-Chair and the 2004 IEEE ICRA Program Chair. He is also the Editor-in-Chief of the IEEE Robotics and Automation Magazine.

Dr. Valavanis is a former Fulbright Scholar.

Nikos C. Tsourveloudis received the B.S. and Ph.D. degrees in production engineering and management from the Technical University of Crete, Chania, Greece, in 1990 and 1995, respectively.

He was President of the Hellenic Association of Industrial Engineers from 1992 to 1995. During the academic years 1997 and 1998, he was a Visiting Research Scientist at the Center for Advanced Computer Studies, CACS, and the Apparel-Computer Integrated Manufacturing Center (A-CIM) of the University of Louisiana at Lafayette, Louisiana, USA. Currently, he is an Assistant Professor at the Department of Production Engineering and Management and Director of the Machine Tools Laboratory at the Technical University of Crete. His research interests are in the areas of intelligent control, modeling and scheduling of flexible and computer integrated manufacturing systems, autonomous operation/navigation of uninhabitant vehicles, virtual reality and fuzzy logic applications.

Dr. Tsourveloudis is a member of numerous professional and scientific organizations. His research has been funded by the NSF, the LEQSF and currently by the Hellenic GSRT.

Anargyros N. Kostaras received the B.S. degree in production engineering and management from the Technical University of Crete, Chania, Greece, in 2001. He is currently a post-graduate student at the London School of Economics, U.K.

His research interests are in the fields of optimization, artificial intelligence and autonomous operation/navigation of unmanned vehicles.