The Aeronautical Journal (2025), **129**, pp. 902–938
doi:10.1017/aer.2024.132

ROYAL AERONAUTICAL SOCIETY

RESEARCH ARTICLE

Genetic algorithm-based path planning of quadrotor UAVs on a 3D environment

M.A. Gutierrez-Martinez[1], E.G. Rojo-Rodriguez[1], L.E. Cabriales-Ramirez[1], K. Estabridis[2] and O. Garcia-Salazar[1]

[1]Aerospace Engineering Research and Innovation Center, Faculty of Mechanical and Electrical Engineering, Autonomous University of Nuevo Leon, Apodaca, Nuevo Leon, Mexico
[2]Naval Air Warfare Center Weapons Division, Research Department, CA, USA
Corresponding author: Garcia-Salazar; Email: octavio.garciasl@uanl.edu.mx

Received: 26 April 2024; **Revised:** 24 August 2024; **Accepted:** 30 October 2024

Keywords: path planning; genetic algorithm ; UAVs; obstacle avoidance

Abstract

In this article, a genetic algorithm (GA) is proposed as a solution for the path planning of unmanned aerial vehicles (UAVs) in 3D, both static and dynamic environments. In most cases, genetic algorithms are utilised for optimisation in offline applications; however, this work proposes an approach that performs real-time path planning with the capability to avoid dynamic obstacles. The proposed method is based on applying a genetic algorithm to find optimised trajectories in changing static and dynamic environments. The genetic algorithm considers genetic operators that are employed for path planning, along with high mutation criteria, the population of convergence, repopulation criteria and the incorporation of the destination point within the population. The effectiveness of this approach is validated through results obtained from both simulations and experiments, demonstrating that the genetic algorithm ensures efficient path planning and the ability to effectively avoid static and dynamic obstacles. A genetic algorithm for path planning of UAVs is proposed, achieving optimised paths in both static and dynamic environments for real-time tasks. In addition, this path planning algorithm has the properties to avoid static and moving obstacles in real-time environments.

Nomenclature

UAV	unmanned aerial vehicles
GA	genetic algorithm
GaCPo	genetic algorithm with population of convergence
GaHM	genetic algorithm with high mutation
GaLM	genetic algorithm with low mutation
n	number of alleles
N_g	number of genes
x_b, y_b, z_b	x, y, z coordinates in binary
$x_{dec}, y_{dec}, z_{dec}$	x, y, z coordinates in decimal
f_i	objective function
$x_{ini}, y_{ini}, z_{ini}$	x, y, z initial point coordinates
x_i, y_i, z_i	x, y, z waypoint coordinates
x_{fi}, y_{fi}, z_{fi}	x, y, z final point coordinates
x_o, y_o, z_o	x, y, z obstacles coordinates
p	penalty
P	size population
pesoxy	number of times that a path cross an obstacle in the plane XY
pesoxz	number of times that a path cross a obstacle in the plane XZ

© The Author(s), 2024. Published by Cambridge University Press on behalf of Royal Aeronautical Society.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

pesoyz	number of times that a path cross a obstacle in the plane *YZ*
W_p	waypoint

Greek symbol

αDI	angle between right and left tangent point
αLI	angle between left tangent point and obstacle
αLD	angle between left tangent point and obstacle

1.0 Introduction

Path planning is an essential task in robotics, representing one of the most complex challenges in computer science. With significant advancements in unmanned aerial vehicles (UAVs) in recent years, these aerial vehicles have found applications in both civilian and military domains. Path planning algorithms have been implemented in UAVs to find paths that align with specific mission conditions within their respective environments [1, 2].

Path planning is a method that finds an obstacle-free path from a starting point to an endpoint within a working environment [3]. Several classical approaches have been explored to address this challenge, including potential fields [4, 5], cell decomposition [6], and visibility graphs [7]. However, owing to the inherent complexity of path planning, this type of problem is classified as an NP-hard optimisation problem [8]. This classification can lead to inefficient classical methods due to high computational costs and the risk of getting trapped in local minima [9].

As a result, heuristic methods and bio-inspired approaches, such as particle swarm optimisation (PSO) [10], neural networks (NN) [11], ant colony optimisation (ACO) [12], and genetic algorithms (GA) [13], have been employed to address the path planning problem in robotics. These methods are capable of circumventing obstacles, maintaining minimal distances, and reducing the risk of getting trapped into local minima [14].

GAs, as proposed by J. Holland in Ref. [15], are metaheuristic techniques inspired by natural processes such as evolution and genetics. GAs operate with populations of individuals, where each individual represents a solution to a specific problem. Genetic operators generate new individuals within the population, eliminating the less suitable ones while preserving the best performers. Simultaneously, the objective function assesses the performance of each individual. In this way, genetic algorithms explore multiple solutions with the goal of discovering the global minimum of the objective function. Due to their effective search capabilities and optimisation approach, GAs prove to be an efficient choice for addressing path planning challenges.

Path planning algorithms have been employed in static environments, where the surroundings are well-known, and obstacles remain stationary. For instance, in Ref. [16], a GA was proposed for static environments, incorporating and enhancing crossover operator to avoid premature convergence. This methodology successfully achieved the shortest distances, safe paths and minimal turns in just a few iterations. Similarly, in Ref. [17], a GA was combined with a prior knowledge method to create paths circumventing radar zones. The prior knowledge method expedited a path discovery, and simulations confirmed the generated paths for UAVs effectively avoided restricted zones. However, the operational reality for UAVs often involves dynamic environments with moving obstacles. Consequently, algorithms capable of adapting to these dynamic conditions have become essential. This implies that algorithms must react in real-time and dynamically search for an optimal path to the final destination. While certain algorithms, such as potential fields, have been utilised for dynamic environments [18, 19], these methods tend to fall into local minima. In response to these challenges, specialised evolutionary methods for dynamic environments have been developed. In Ref. [9], a hybrid algorithm combining a GA with fuzzy logic was introduced for finding paths in dynamic 2D environments. This approach significantly improved computation times compared to those using the algorithms in isolation. Furthermore, in

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

Ref. [20], an enhanced genetic algorithm called GADPP was employed for dynamic environments considering factors such as path length, time and Bezier curves, leading to a reduction in the time required to obtain optimised paths. Another notable example is presented in Ref. [21], where parallel GAs were utilised to address real 3D environments. The objective function incorporated UAV dynamics, and the proposed algorithm demonstrated efficiency owing to the parallelisation of the objective function.

In the study presented by Mohamed [22], a modified genetic algorithm (MGA) was utilised to derive Bezier curves for path planning in dynamic environment. The MGA allowed for a more diverse exploration, generating solutions that, when implemented, resulted in smooth paths. These Bezier curves minimised distance and, consequently, conserved energy in the vehicle. In the work of Shivgan [23], a GA was proposed to minimise the energy consumption when solving the traveling salesman problem for UAVs. This algorithm successfully reduced both the number of turns and the overall distance traveled in the paths. The results demonstrated a 2.5 times reduction in energy consumption by minimising turns. Arantes et al. [24], developed a hybrid approach, combining a multi-population GA with a visibility graph to find UAV paths in a non-convex environment with uncertainties. This hybrid method efficiently discovered paths within a maximum of 10 seconds. Volkan et al. [25] addressed path planning for autonomous UAVs using a combination of a genetic algorithm, ant colony optimiser, Voronoi diagram and clustering methods. Suboptimal paths were implemented through the ant colony optimiser and integrated into an initial GA population, achieving a 70% reduction in required objective function evaluations. In Ref. [26], a new GA called the multi-frequency vibrational genetic algorithm (mVGA) was introduced for solving path planning problems in different 3D environments for UAVs. The algorithm featured a new mutation strategy to increase diversity and utilise clustering strategies along with the Voronoi diagram in the initial population. Chaoqun et al. [27] applied a heuristic strategy, combining heuristic crossover with a SAR algorithm (search and rescue optimisation algorithm) to enhance convergence speed and maintain diversity in the population. This strategy allowed real-time adjustments to paths, straightening the flight path of UAVs. In the work proposed by Pan et al. [28], a deep learning genetic algorithm (DL-GA) was proposed for rapid path optimisation, leveraging the advantages of both methods. Experiments confirmed that DL-GA significantly accelerated the path solving for UAVs. Chen et al. [29] developed a genetic algorithm-based path planning algorithm for UAVs in autonomous cruise operations, ensuring the selection of efficient and reliable paths in complex environments. The experiments revealed that the proposed algorithm found shorter paths compared to traditional GAs.

The main contribution of this work is the development of a genetic algorithm for path planning in static and dynamic environments applied to quadrotor UAVs. A population of convergence criterion with a high mutation rate is proposed to maintain diversity while ensuring convergence. Additionally, our algorithm includes a repopulation criterion to preserve population diversity and prevent sticking to a reference point, along with a criterion to introduce the target point into the convergence population. These criteria provide solutions in a few iterations for static environments and to avoid mobile obstacles in dynamic environments. Simulations are conducted using an identified quadrotor model to validate the proposed path planning algorithm, and the experimental results support the effectiveness of our approach. The main contributions are summarised as follows:

- A genetic algorithm for path planning of UAVs is proposed achieving optimised paths in 3D motion, static and dynamic environments.
- The proposed genetic algorithm for path planning of UAVs is able to avoid static and mobile obstacles.
- The proposed genetic algorithm is implemented in numerical simulations and real-time experiments using quadrotor UAVs.

The organisation of this work is as follows: Section 2 presents the problem statement. Section 3 details the design of the genetic algorithm, including the criteria used for the objective function to minimise distance and avoid obstacles, as well as the genetic operators for selection, crossover, mutation and the improved criteria. Section 4 presents the results of the algorithm characterisation, including a case in a

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

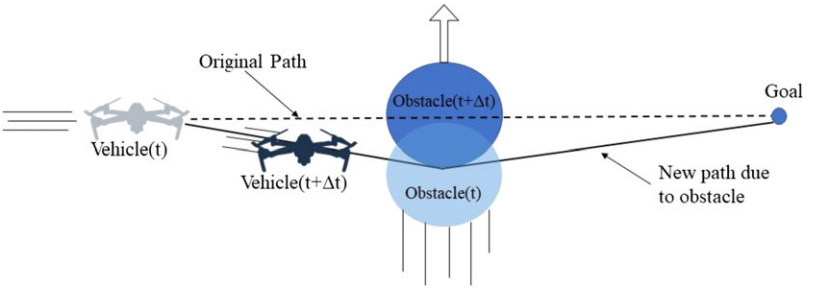


Figure 1. *Algorithm operation in dynamic environment.*

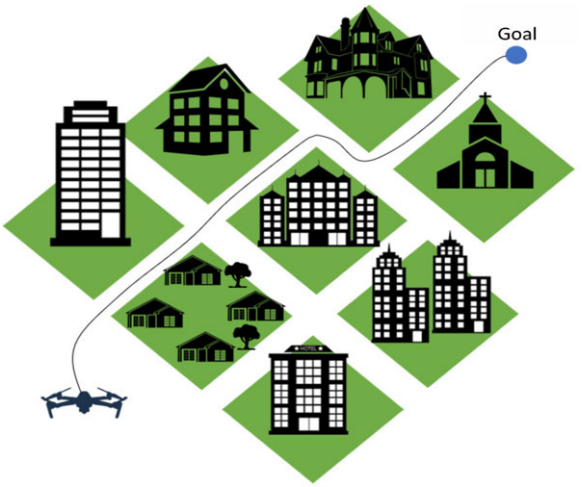


Figure 2. *Algorithm operation in static environment.*

static environment and the proposed cases for dynamic environments. In addition, identified model is described and the test of the path planning algorithm with numerical simulations which specifies the conditions for the real-time tests. The conclusions are presented in Section 5.

2.0 Problem statement

The path planning algorithm of UAVs considering the obstacle avoidance in a 3D environment is addressed in this research work. Indeed, the path generation enables aerial vehicles to autonomously navigate avoiding static and dynamic obstacles in a 3D environment so that this design can apply a single aerial vehicle or a group of aerial vehicles.

Our proposed genetic algorithm is designed to discover paths that satisfy the minimum distance requirement and avoid obstacles, even in dynamic settings where obstacles are in continuous motion. For this purpose, enhanced criteria is employed, including a high mutation rate, repopulation, a convergence population and the integration of the final destination point within the convergence population. Then, paths are identified and are adapted to unforeseen changes caused by obstacles, as depicted in Fig. 1. Furthermore, our algorithm is utilised to determine optimised paths within static environments, as demonstrated in Fig. 2, particularly in urban areas, regions with buildings, residential zones and other locations where avoiding restricted areas is essential.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

Remark 1. A genetic algorithm for the path planning of UAVs is proposed, achieving optimised paths to avoid static and dynamic obstacles in a 3D environment. For design purposes, this algorithm considers obstacles or objects as 3D spheres when UAVs navigate around the contours of these objects. Thus, the UAVs fly tangentially around the objects according to the trajectory generated by the genetic algorithm in real-time.

3.0 Collision avoidance algorithm

This section presents a novel strategy employed by the proposed algorithm to avoid static and dynamic obstacles in a 3D environment. The strategy is based on improved criteria for population and repopulation convergence, incorporating high mutation rates and a waypoint at the final point. Furthermore, a multi-objective function is introduced to minimise the path and avoid obstacles. Finally, the genetic operators and their specifications aimed at minimising the path are described.

3.1 Initialisation of population and chromosome encoding

Our proposed methodology establishes a random initial population of individuals and genes.

Definition 1. *Let the matrix Po_i be defined by $m \times n$ where m is the individuals and n is the alleles with $\in A = \{0, 1\}$,*

$$o_i = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \tag{1}$$

The matrix Po_i can be decomposed into three submatrices, denoted as sub_1, sub_2, and sub_3, storing information related to the coordinates X, Y and Z, respectively.

Definition 2. *Let the matrices sub_1, sub_2, $sub_3 \in Po_i$ be defined by $m \times k$ where m is the population size, $k = n/N_g$, n in the number of alleles, and N_g is the number of genes for each individual.*

$$Po_i = \begin{pmatrix} [sub_1] & [sub_2] & [sub_3] \end{pmatrix} \tag{2}$$

In our study, a random initial population of 60 individuals and 24 alelles is defined; thus, matrices sub_1, sub_2, sub_3 are proposed as

$$[sub_1] = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,8} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,8} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,1} & a_{60,2} & \cdots & a_{60,8} \end{pmatrix} \tag{3}$$

$$[sub_2] = \begin{pmatrix} a_{1,9} & a_{1,10} & \cdots & a_{1,16} \\ a_{2,9} & a_{2,10} & \cdots & a_{2,16} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,9} & a_{60,10} & \cdots & a_{60,16} \end{pmatrix} \tag{4}$$

$$[sub_3] = \begin{pmatrix} a_{1,17} & a_{1,18} & \cdots & a_{1,24} \\ a_{2,17} & a_{2,18} & \cdots & a_{2,24} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,17} & a_{60,18} & \cdots & a_{60,24} \end{pmatrix} \tag{5}$$

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

where $x_b = sub_1$, $y_b = sub_2$ and $z_b = sub_3$ which x_b, y_b and z_b contain the binary coordinates x, y and z, respectively. The phenotype of x_b, y_b and z_b are denoted as x_{dec}, y_{dec} and z_{dec}, representing their decimal values as real numbers in the range of $0 \leq x_{dec}, y_{dec}, z_{dec} \leq 255$. Finally, these concepts are extended to the entire population Pob_{dec} and a conversion factor f_c is utilised to scale the three-dimensional space as established in Equation (6).

$$Pob_{dec} = \left([x_{dec}] /f_c \quad \left[y_{dec}\right] /f_c \quad [z_{dec}] /f_c \right) \tag{6}$$

The proposed genetic algorithm operates with binary populations, and the search resolution depends on the length of the chromosome chain and the population size. The chromosome is divided into three genes $N_g = 3$. In our proposed algorithm, 8-bit chains or alleles are used to represent each gen X, Y and Z coordinate in space, and genetic operators are employed to generate new individuals.

3.2 Objective functions

The objective function is used to assess the quality of path, and a multi-objective function, which considers three criteria, is utilised.

Definition 3. *Let the multi-objective function f_i be defined by three objective functions referred to as criteria*

$$\min f_i = \min \left(f_{1_i} + f_{2_{i,j}} + f_{3_{i,j}}\right) \tag{7}$$

where f_{1_i} is the criterion of the length of path, $f_{2_{i,j}}$ is the criterion of distance between a waypoint and an obstacle, and $f_{3_{i,j}}$ is the criterion of the sum of path angles with i is the number of individuals and j is the number of obstacles.

The first criterion is the length of the path for obtaining the minimum distance; the second is the minimum distance between waypoints and obstacles to determine the waypoints unfeasibly; and the third is the sum of path angles to determine if the path crosses an obstacle. Thus, the genetic algorithm minimises the objective function f_i.

3.2.1 Length of path

The first criterion of the multi-objective function calculates the distance between the initial point, the waypoint, and the final point, as shown in Fig. 3. All paths are evaluated, and consequently, the longest distance receives the highest fitness value, reducing its likelihood of reproduction. In Algorithm 1, in the section on objective function 1, you can find the corresponding pseudocode for the objective function. Equation (7) calculates the distance between the initial point, the waypoint and the final point.

$$f_{1_i} = \sqrt{(x_{ini} - x_i)^2 + (y_{ini} - y_i)^2 + (z_{ini} - z_i)^2} + \sqrt{(x_i - x_{f_i})^2 + (y_i - y_{f_i})^2 + (z_i - z_{f_i})^2} \tag{8}$$

3.2.2 Distance between waypoint and obstacles

The second criterion penalises the paths where the waypoints are in an obstacle, eliminating these paths in the first iterations. Thus, this criterion prevents the genetic operators from working with inefficient paths in subsequent iterations. This criterion eliminates waypoints unfeasible, allowing quick convergence, which helps in a dynamic environment. The second criterion of the multi-objective function is shown in algorithm 1 in objective function 2. As shown in Fig. 4, the distance between the waypoint and the obstacle is determined. Then, waypoints are evaluated; if the distance is less or equal to the radius of the obstacle, waypoints are inside the obstacle, and a penalty is added to its fitness value.

A strategy of the two-dimensional plane is employed, as shown in Fig. 5. Calculations are made with projections of obstacles and waypoints on the planes XY, YZ and XZ. Equations (9)–(11) calculate the

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

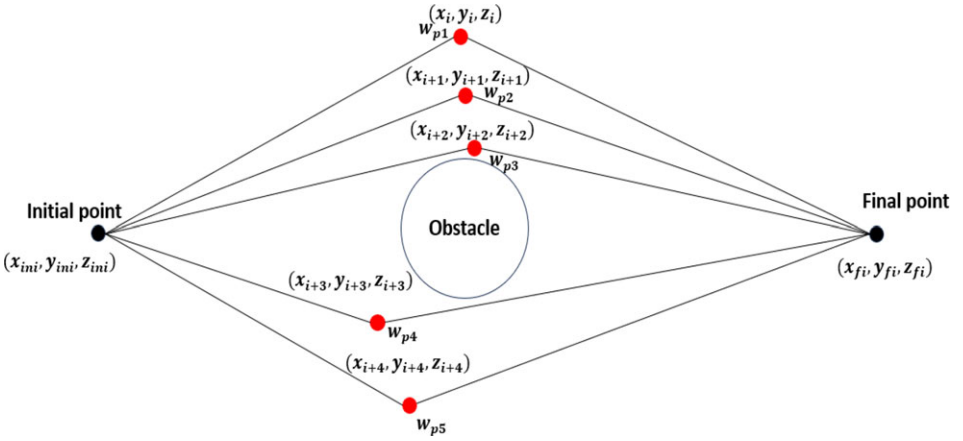


Figure 3. *Evaluation of the path length.*

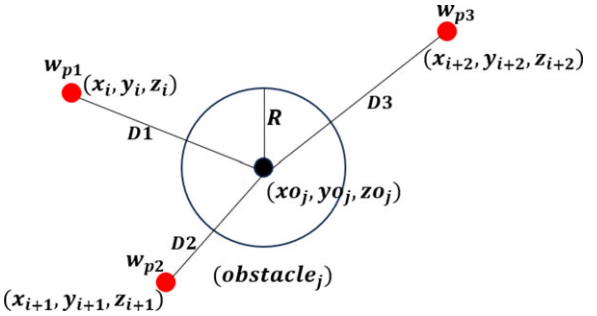


Figure 4. *Evaluation of the waypoint.*

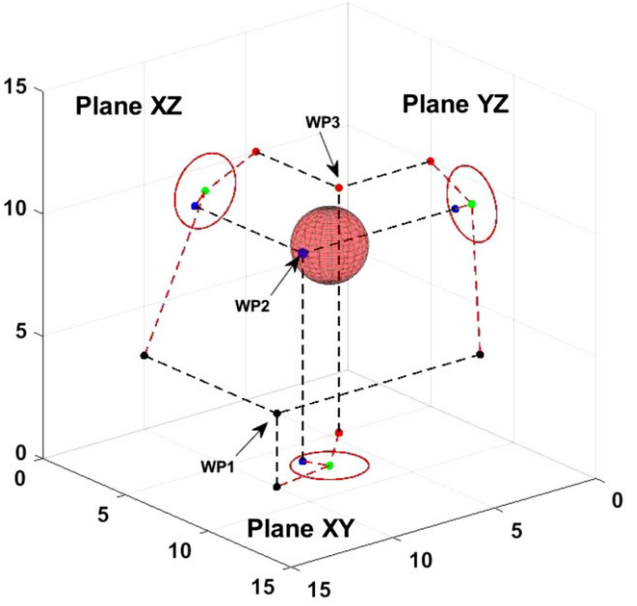


Figure 5. *Evaluation of the 2D planes.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

distance of obstacles in the different planes, where xo_j, yo_j, zo_j correspond to the coordinates of the obstacle.

$$doxy_i = \sqrt{(x_i - xo_j)^2 + (y_i - yo_j)^2} \tag{9}$$

$$doxz_i = \sqrt{(x_i - xo_j)^2 + (z_i - zo_j)^2} \tag{10}$$

$$doyz_i = \sqrt{(y_i - yo_j)^2 + (z_i - zo_j)^2} \tag{11}$$

The Equation (12) defines the second objective function, where f_{2_i} is initialised to 0, and the equation involves a sum of the Equation (13).

$$f_{2_i} = \sum_{j=1}^{numobs} fs_{2_{i,j}} \tag{12}$$

If all three distance conditions relative to the obstacle are fulfilled, the waypoint is considered within the obstacle, and a penalty is applied. However, if any of the distance conditions is less than the radius of the obstacle, no penalty is added, as demonstrated in Equation (13).

$$fs_{2_{i,j}} = \begin{cases} fs_{2_{i,j}} + p, & if \quad doxy_i \text{ and } doxz_i \text{ and } doyz_i \leq radius \\ fs_{2_{i,j}} + 0, & if \quad doxy_i \text{ or } doxz_i \text{ or } doyz_i > radius \end{cases} \tag{13}$$

3.2.3 Sum of angles of the path

The third criterion is presented in algorithm 1 within objective function 3, assessing whether the path intersects with the obstacle. Calculations are made to determine whether an angle intersects an obstacle involving the angle formed by the path and the virtual vectors of the right and left tangent points. The evaluation of the sum of angles for the path criterion occurs in two segments: initially between the initial point and the waypoint, as illustrated in Fig. 6, followed by the segment between the waypoint and the final point, as shown in Fig. 7. Equation (14) is utilised to compute the angle among the tangent points, denoted as angleDI (αDI). The angle between the path and the virtual vector on the right is derived using Equation (15), referred to as angleLI (αLI). Lastly, the angle with the virtual vector on the left is computed using Equation (16), known as angleLD (αLD).

$$\alpha DI_{i,j} = \arccos \frac{\overrightarrow{U_{i,j}}.\overrightarrow{V_{i,j}}}{|U_{i,j}|\,|V_{i,j}|} \tag{14}$$

$$\alpha LI_{i,j} = \arccos \frac{\overrightarrow{U_{i,j}}.\overrightarrow{W_{i,j}}}{|U_{i,j}|\,|W_{i,j}|} \tag{15}$$

$$\alpha LD_{i,j} = \arccos \frac{\overrightarrow{W_{i,j}}.\overrightarrow{V_{i,j}}}{|W_{i,j}|\,|V_{i,j}|} \tag{16}$$

The calculation of the angles (αLD), (αLD) and (αLD) is performed on a 2D plane to reduce processing and computation time. This process is carried out in the space XY, XZ and YZ. The number of times a path crosses an obstacle is saved in a variable $pesoxy_{i,j}$, $pesoxz_{i,j}$ and $pesoyz_{i,j}$ for each plane. The Equation (17) shows the conditions to determine if a path crosses an obstacle. The Equation (18) states that $pesoxy_{i,j}$, $pesoxz_{i,j}$ and $pesoyz_{i,j}$ only increase when all three variables detect that the path crosses an obstacle; if any of the variables equal 0, it means that the path does not cross the obstacle. On the

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

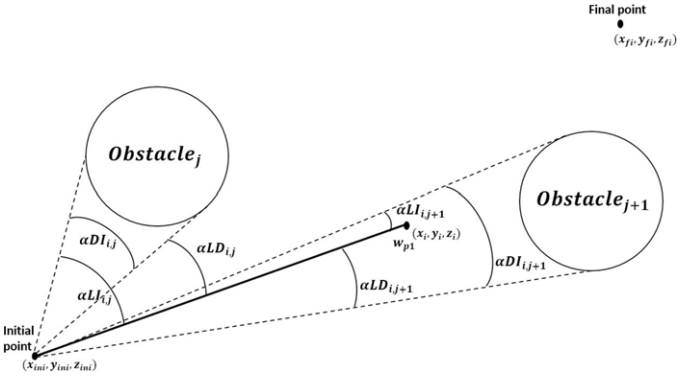


Figure 6. *Evaluation of the path crossing by obstacles 1.*

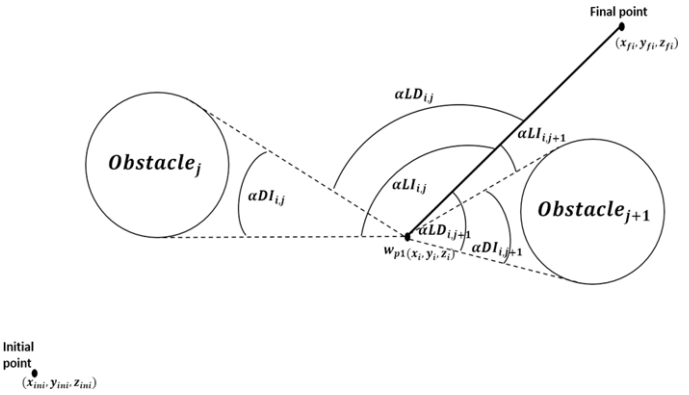


Figure 7. *Evaluation of the path crossing by obstacles 2.*

other hand, Equation (19) defines the criterion 3 penalising the number of times that a path crosses an obstacle where c is a constant of penalisation. This value is multiplied by a constant and added to the fitness value.

$$\begin{bmatrix} pesoxy_{i,j} \\ \cdots \\ pesoxz_{i,j} \\ \cdots \\ pesoyz_{i,j} \end{bmatrix} = \begin{cases} pesoxy_{i,j}+1, & if \quad \alpha DIxy_{i,j} = \alpha LDxy_{i,j} + \alpha LDxy_{i,j} \\ pesoxy_{i,j}+0, & if \quad \alpha DIxy_{i,j} \neq \alpha LDxy_{i,j} + \alpha LDxy_{i,j} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ pesoxz_{i,j}+1, & if \quad \alpha DIxz_{i,j} = \alpha LDxz_{i,j} + \alpha LDxz_{i,j} \\ pesoxz_{i,j}+0, & if \quad \alpha DIxz_{i,j} \neq \alpha LDxz_{i,j} + \alpha LDxz_{i,j} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ pesoyz_{i,j}+1, & if \quad \alpha DIyz_{i,j} = \alpha LDyz_{i,j} + \alpha LDyz_{i,j} \\ pesoyz_{i,j}+0, & if \quad \alpha DIyz_{i,j} \neq \alpha LDyz_{i,j} + \alpha LDyz_{i,j} \end{cases} \tag{17}$$

$$f_{3_i} = \begin{cases} fs_{3_i}, & if \quad pesoxy_{i,j} \text{ and } pesoxz_{i,j} \text{ and } pesoyz_{i,j} \geqslant 1 \\ 0, & if \quad pesoxy_{i,j} \text{ or } pesoxz_{i,j} \text{ or } pesoyz_{i,j} < 1 \end{cases} \tag{18}$$

$$fs_{3_i} = \sum_{j=1}^{numobs} pesoxy_{i,j} \cdot c + \sum_{j=1}^{numobs} pesoxz_{i,j} \cdot c + \sum_{j=1}^{numobs} pesoyz_{i,j} \cdot c \tag{19}$$

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

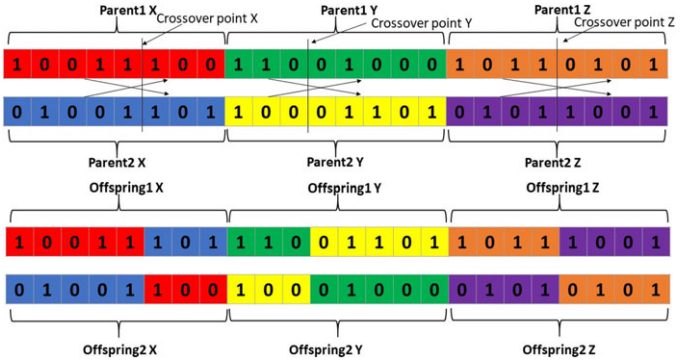


Figure 8. *Crossover genetic operator.*

3.3 Genetic operators

The parameters of genetic operators are chosen to have a fast convergence without falling into an inefficient solution so that the genetic algorithm can make quick decisions against unexpected changes; this is required in a dynamic environment. An elitist selection strategy is used, in which, although a rapid convergence is reached, falling in a local minimum is avoided due to wide sampling resolution and the high percentages of crossover and mutation that further increase the resolution sampling. Since a high percentage of alteration in the population is considered, the convergence population is executed to guarantee convergence with a high mutation criterion. This convergence population is a part of the population that is not affected by the mutation.

3.3.1 Selection operator

The first genetic operator after the individuals are evaluated by the objective function, which is the natural selection operator. This operator selects the individuals that best meet the conditions based on their fitness value. The best individuals are those who meet the condition for which $pesoxy_{i,j}$, $pesoxz_{i,j}$ and $pesoyz_{i,j}$ equal 0, and its path length is the shortest. However, individuals that do not meet the boundary can be selected to maintain diversity in the population. A natural selection operator with elitist criteria is utilised; this allows ordering individuals from least to greatest fitness value. This operator selects the half of the population with the lowest fitness value to reproduce these individuals in the crossover operator.

3.3.2 Crossover operator

The crossover operator creates new solutions for the problem. This operator is used to reproduce the individuals selected by natural selection. Each crossover occurs only between each gene; each axis has a single point of crossover so that the crossover only happens between them. The crossover point varies randomly in each iteration, and the pairing process of parents, in turn, is random, as shown in Fig. 8. A crossover rate of 100% is used for the population selected by the crossover operator. This means that the entire population experiences crossover.

3.3.3 Mutation operator

The mutation helps to preserve the diversity of the population and prevent premature convergence. Similar to the crossover operator, the mutation occurs randomly by altering the chain of bits in the

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

Algorithm 1: Genetic algorithm for path planning.

```
Set initial point x_ini ,y_ini ,z_ini and final point x_fi ,y_fi ,z_fi
Set variable P population size, N_g number of genes.
Set Pc crosrover rate,Pm Mutation rate
Set numobs as number of obstacles.
Set counter i = 1, j = 1, k = 1
Set radius as radio of obstacles.
Set numway as number of waypoints.
Initializes population
while UAV moves to the final point do
    while i ≤ P do
        Calculate the distance f_1i by (7)                    //Objective function 1
        while j ≤ numobs do
            Calculate distance between the waypoints and obstacles by
            doxy_i eq. (9), doxz_i eq. (10), doyz_i eq. (11)   //Objective function 2
            if doxy_i & doxz_i & doyz_i V radius then
                add penalty by eq. (12)
            else
                do not add penalty
            end
            while k ≤ numway + 1 do
                get angle αDI_i,j by eq. (14)                  //Objective function 3
                get angle αLI_i,j by eq. (15)
                get angle αLD_i,j by eq. (16)
                AngleSum = αLD_i,j + αLI_i,j
                if αDI_i,j = AngleSum then
                    peso_k = 1;
                else
                    peso_k = 0;
                end
                k = k + 1
            end
            j = j + 1
        end
        i = i + 1
    end
    Selection operator
    Crossover operator
    Mutation operator
    Criteria for dynamic environment
    Replace population
end
```

population. The alleles, affected by the mutation, are calculated by Equation (20), where *P* represents the number of individuals and *n* represents the number of alleles.

$$Tgenes = P \cdot n \tag{20}$$

$$Genesmut = Tgenes \cdot prob/100 \tag{21}$$

Algorithm 1 presents the genetic algorithm with the objective functions and genetics operators.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

3.4 Criteria for static and dynamic environment

The proposed algorithm is described considering the criteria for the dynamic environment in which the ability to recalculate the path when a convergence occurs in the population is essential. When the waypoint has converged, and an obstacle has obtained in the way, it is necessary to have diversity in the population to calculate a new path. Thus, a high mutation strategy is used to avoid getting stuck at the waypoint to solve this problem. A percentage of 2% is used, which means the 50% percentage of the population is altered in each iteration. This provides diversity; however, high mutation can suddenly alter the waypoint followed by the UAV during motion. Thus; to solve this problem, a new criterion of convergence population is used, in such a way the first 25% percentage individuals of the population are not affected by the mutation. This criterion, called population of convergence, allows for high mutation rates without losing convergence maintaining the best waypoint in the population with high diversity. Then, when an individual is a suitable solution, this enters the population of convergence and leaves out another individual of less quality; in this way, the population of convergence also can be improved.

After a UAV avoids an obstacle, it may not move since the waypoint has converged and there is no diversity in the population. Thus, to solve this problem, a new random population is created, providing diversity in the population, and a new waypoint is chosen for the aerial vehicle. The algorithm introduces a new random population when the distance between the UAV and the waypoint is less than a set distance (*distr*). This way, 50% percent of the population is created again when the UAV is near the waypoint.

When the vehicle has avoided all obstacles, it can get stuck in the waypoint while finding a new waypoint to reach the final point. Thus, calculating a new path can take time and cause problems. This criterion allows the UAV to proceed to the final point when there is no obstacle between the path calculated and the final point. For this purpose, the coordinates of the final point are determined, and then the coordinates in their decimal value are converted to binary and introduced into the convergence population. This path is unaffected by the high mutation, and if no obstacle interferes with the path, the final point always predominates in the convergence population. It is chosen as the best waypoint for the population. Algorithm 2 shows the proposed approach.

4.0 Result

4.1 Platform setup

The equations for generating the continuous path are shown by Equations (22)–(24), $x_d\,(t)$, $y_d\,(t)$, $z_d\,(t)$ are the desired position, t is the time; t_a is an earlier time, and x_0, y_0, z_0 is the initial position.

$$x_d\,(t) = m_x \cdot (t - t_a) + x_0 \tag{22}$$

$$y_d\,(t) = m_y \cdot (t - t_a) + y_0 \tag{23}$$

$$z_d\,(t) = m_z \cdot (t - t_a) + z_0 \tag{24}$$

where

$$m_x = \left(x_f - x_0\right) / \left(t_f\right) \tag{25}$$

$$m_y = \left(y_f - y_0\right) / \left(t_f\right) \tag{26}$$

$$m_z = \left(z_f - z_0\right) / \left(t_f\right) \tag{27}$$

The x_f, y_f, and z_f are the coordinates of the waypoint given by the genetic algorithm, $t_f = newdist/vel$ is the time final, *newdist* is the new distance of the path if there is a change, *vel* is the velocity of UAV. The final time is recalculated when the waypoint used for the navigation changes; this maintains the

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

Algorithm 2: Criteria for static and dynamic environment.

Set $distr$ as distance limit between initial point and waypoint.
Set Variable n as number of alleles
Set Variable f_c as conversion factor
Set constant p_{safe} as number of population of convergence
$Tgenes = P \cdot n$;
$GenesMut = Tgenes \cdot prob/100$
while $i \leq GenesMut$ **do**
  $fila = randi([1, P])$
  $colum = randi([1, n])$
  if $fila \geq P_{safe}$ **then**
    $Pob(fila, colum) = 1 - Pob(fila, colum)$;
  end
  $i = i + 1$
end
Calculate $dist$ between Initial point and waypoint eq. (7)
if $dist \leq distr$ **then**
  $Pob(P/4 + 1 : P, 1 : n) = round(rand(P/4, n))$;
end
$x_{f_i}, y_{f_i}, z_{f_i}$ //Final point
$x_c = x_{f_i}/f_c, y_c = y_{f_i}/f_c, z_c = z_{f_i}/f_c$
$x_b = dec2bin(x_c), y_b = dec2bin(y_c), z_b = dec2bin(z_c)$ //Convert to binary
$Pob(5, 1 : n) = [x_b, y_b, z_b]$ //enter the final point in the population of convergence

constant velocity of the UAV. In addition, only recalculated t_f and therefore the components of velocity m_x, m_y, m_z, when the waypoint change occurs, as long as there is no change, the values of the components are preserved.

Numerical simulations are carried out to test the proposed algorithm in LabVIEW software using a computer with a Core i7-8750H processor, an Nvidia GTX-1060 GPU, and 16 GB of DDR4 RAM memory at 2,400MHz. For experimental tests, a Crazyflie drone [30, 31] is used, and the transfer functions of drone is obtained using system identification techniques based on experimental data; Equation (28) for the dynamics X, Y and Equation (29) for the dynamics Z.

$$G(s) = \frac{0.9549}{s^2 + 1.229s + 0.9324} \tag{28}$$

$$G(s) = \frac{1850}{s + 1896} \tag{29}$$

Figure 9 describes, in general, the proposed algorithm for the 3D path planning for quadrotor UAVs. The 3D path planning generator is based on a genetic algorithm executed in a computer as a ground station. The 3D path planning generator consists of a genetic algorithm where the objective function evaluates the UAV position, the environment conditions (obstacles), and the population in decimal Pob_{dec} (waypoints). Then, the genetic operators create new individuals and maintain the diversity of the population. Thus, the population is replaced, generating the trajectory. The quadrotor runs a PID controller in real-time and tracks the UAV references generated by the trajectory generator. Note that this proposed approach is implemented for one and multiple UAVs (swarms).

Real-time experiments are carried out in the laboratory with a motion capture VICON system and Crazyflie 2.1 drones, Fig. 10, which are employed as Crazyswarm [32].

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

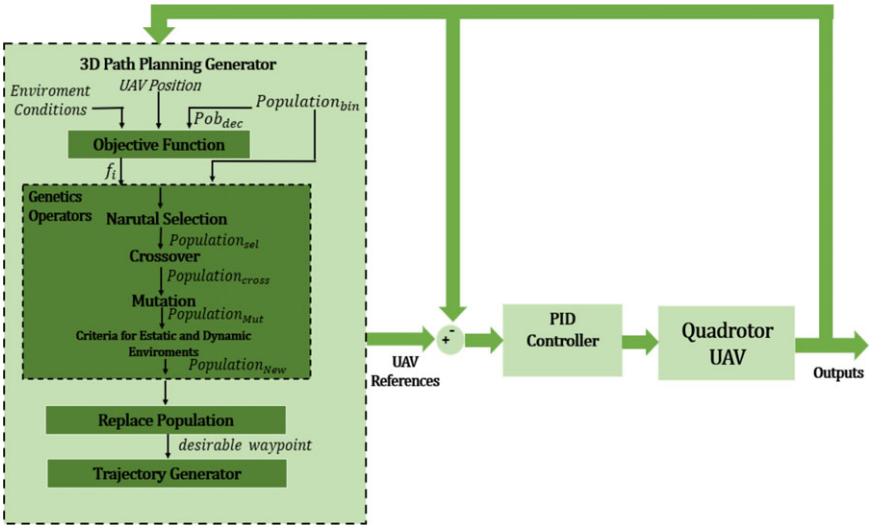


Figure 9. *System diagram.*



Figure 10. *Crazyflie 2.1 drone.*

4.2 Algorithm characterisation in static environment

A case with four obstacles between the initial point (0, 0, 0) and the final point (15, 15, 15) is performed in a static environment to evaluate the capacity of finding the minimum local in the less number of iterations. Table 1 shows obstacle specifications. A comparison between three configurations of GA is carried out; the first configuration is a GA with a population of convergence and high mutation (GaCPo), the second one is a Ga with high mutation but without the population of convergence (GaHM), and the last configuration is a Ga conventional with low mutation (GaLM), which is conventional configuration. For the configuration, GaCPo, GaHM and GaLM have used a percent of mutation 2%, 2% and 0.2%, respectively. Thus, these characteristics are evaluated in a static environment. Figure 11 shows the minimum fitness value obtained, and Fig. 12 shows the number of iterations. Note that a limit of 30 iterations is set for the tests.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

Table 1. *Case static enviroment*

Case static environment		
No. Obs	Position [x(m), y(m), z(m)]	Radio (m)
1	7.5,7,5,7,5	1.5
2	8.5,8.5,8.5	1.5
3	9.5,9.5,9.5	1.5
4	10.5,10.5,10.5	1.5

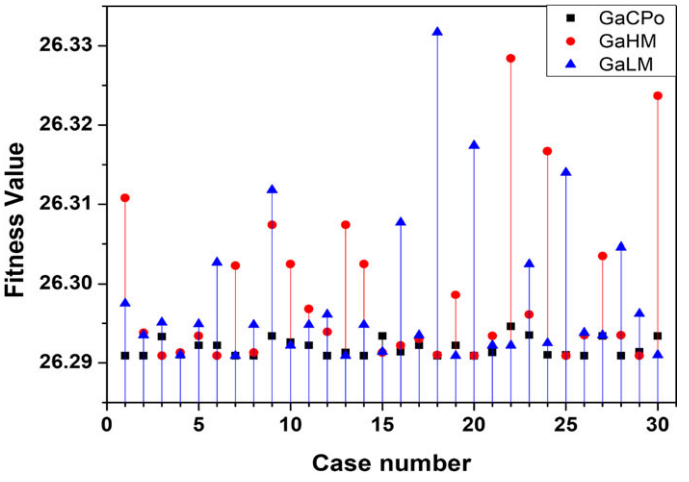


Figure 11. *Fitness value for GaCPo, GaHM and GaLM.*

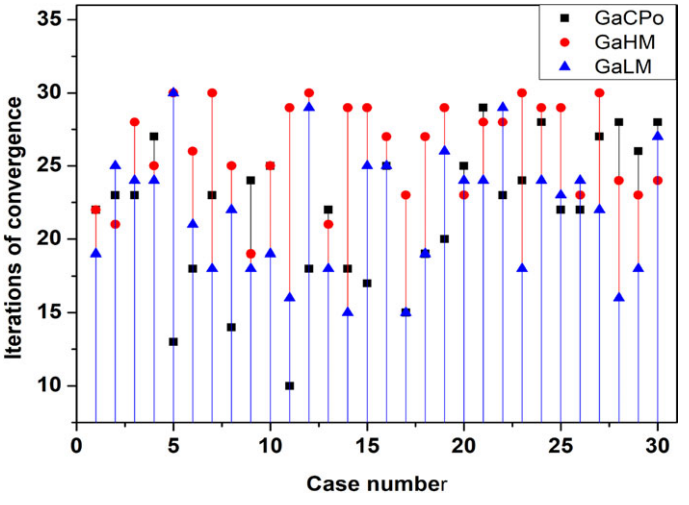


Figure 12. *Iterations of convergence for GaCPo, GaHM and GaLM.*

In the 30 cases executed for the three configurations, it can be observed that the configuration that finds the lowest fitness value on more occasions is the configuration with the population of convergence, in addition to obtaining the lower fitness value compared to the other two configurations. Figure 13 shows that GaCPo obtained the lower average in terms of fitness value with averages of 30 proofs. On the other

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

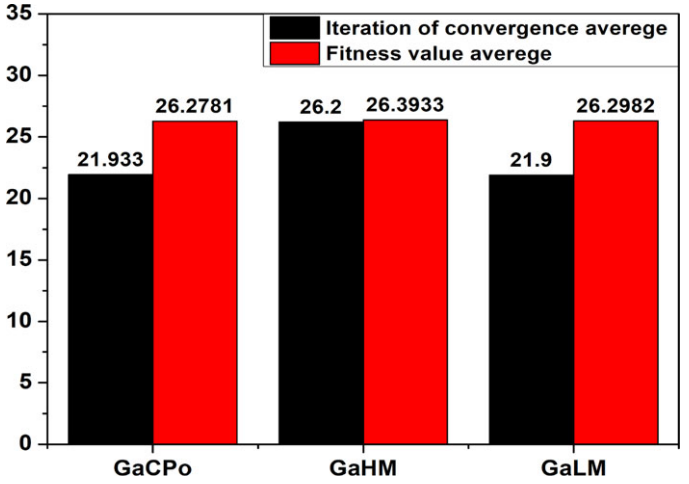


Figure 13. *Avarege for GaCPo, GaHM and GaLM.*

hand, the average of iterations is greater in GaCPo than in the configuration with low mutation GaLM. As expected, the configuration that obtained the best average in convergence is the configuration with the low mutation, but this one also obtained the highest fitness values. In contrast, the configuration with high mutation GaHM, but without the population of convergence, obtained the worst values of iterations; this is due to high mutation making convergence difficult, even in some cases, it changes its fitness value to a higher value.

4.3 Simulations

Some remarks are clarified for the tests in a dynamic online environment.

Remark 2. GaLM is the standard configuration of a genetic algorithm, where mutation rates are around 0.2%. For path planning, this type of configuration is used in offline environments where convergence time is not critical. This configuration is efficient for static cases; however, in a dynamic online environment, convergence might prevent finding new paths in response to changes in the environment.

Remark 3. GaHM provides a high level of population diversity, enabling the algorithm to adapt better to environmental changes caused by obstacles. However, it can also cause unwanted sudden changes in the path.

Remark 4. GaCPo was the strategy used for dynamic online cases, in which a high mutation and a population convergence are obtained. In this approach, part of the population preserves the best individuals for when they are needed. This strategy prevents sudden changes in the direction of the path and ensures diversity in the population.

For the validation of the algorithm, numerical simulations are carried out considering four obstacles with a radio of 1.5m and in a space of 15m $\times$ 15m $\times$ 15m; the UAV flies with a constant velocity of 0.5m/s and the obstacles shift in linear and angular motions.

4.3.1 Case 1

Figures 14 and 15 show the generated 3D path with static obstacles from two perspectives. Although the obstacles are static, the path is solved online while the UAV is in motion. Figure 14 shows that the algorithm generated the path below obstacles 1 and 2. It can be observed that the generated path avoids all the obstacles and that there are no changes in the path, which are static obstacles. In addition, the path passes tangentially for obstacle 2. To find a path with a minimum distance in static cases, the path

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

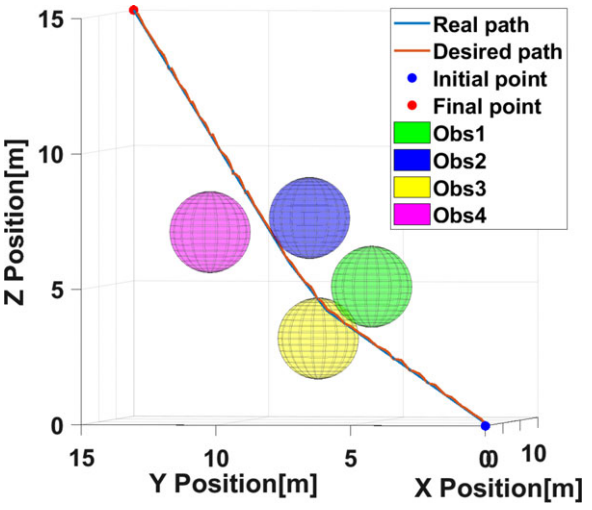


Figure 14. *Simulation: 3D path static obstacles, case 1 and view 1.*

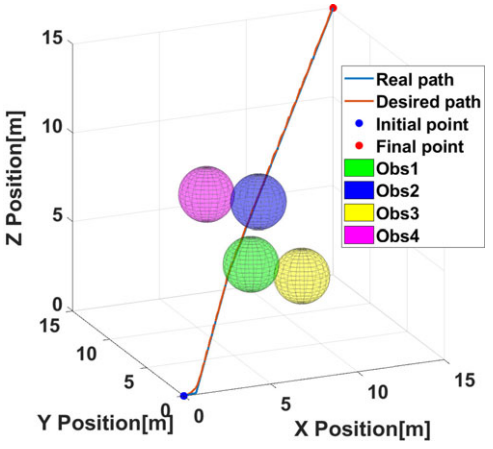


Figure 15. *Simulation: 3D path static obstacles, case 1 and view 2.*

must be tangential to a minimum obstacle. Figures 16, 17 and 18 present the motion of the axes X, Y and Z with respect to time, as well as the real path and the desired path. The real path is made by the simulated model and the control, while the desired path is made by GA and the equations of motion. Figure 19 shows the fitness value of the algorithm, and this graph is important because it shows the behaviour of the path as the UAV moves. It is noted that the fitness value of the entire population has higher fitness value peaks, while the fitness value of the convergence population has smaller peaks as it is from a smaller population and as the individuals have a lower fitness value. It can be seen that between 20 and 30 seconds, there is a slight increase in the fitness value of the population; this is because there is a change in the path to avoid obstacle 2. When the UAV is close to an obstacle and at the final point, there will be some peaks in the fitness value; this is due to the repopulation criterion that helps to have diversity in the population for a change in the path is needed. After 30 seconds, the UAV has avoided all obstacles, and there are no more obstacles in the way of the final point. Hence, the distance of the path becomes the only criterion of the algorithm until the fitness value descends to zero. The descent in the fitness value is presented because the total population and the convergence population begin to converge with individuals that meet the conditions of avoiding obstacles.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

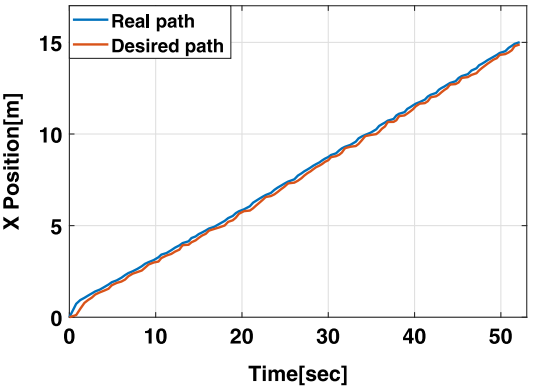


Figure 16. *Simulation: single X-axis, case 1.*

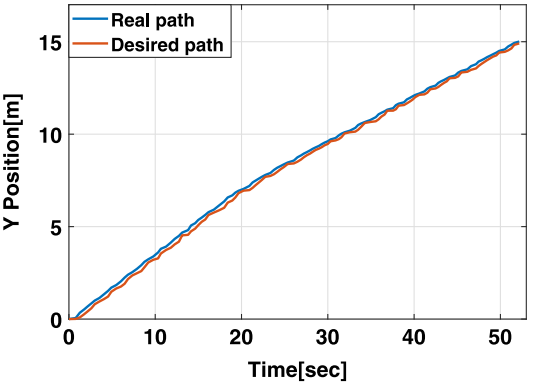


Figure 17. *Simulation: single Y-axis, case 1.*

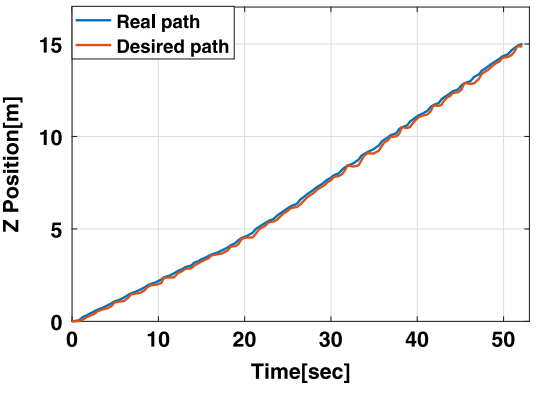


Figure 18. *Simulation: single Z-axis, case 1.*

4.3.2 Case 2

Figures 20 and 21 show the generated 3D path. In this case, obstacles 1 and 2 have a continuous motion; obstacle 1 has an ascending circular motion, and obstacle 2 has an ascending linear motion. It is noted that obstacle 1 is the first to get into the path, then the algorithm makes a slight turn in the first seconds to avoid the collision; this is shown in Figs 22, 23 and 24. This change, in turn, is reflected in the fitness value, Fig. 25, where it can be seen that there is a high peak in the first 10 seconds. This peak is even

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

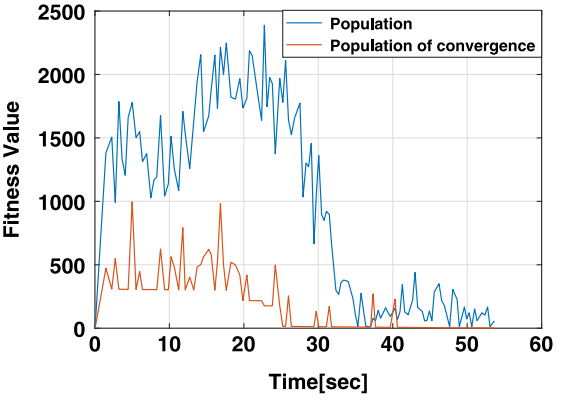


Figure 19. *Simulation: fitness value, static case 1.*

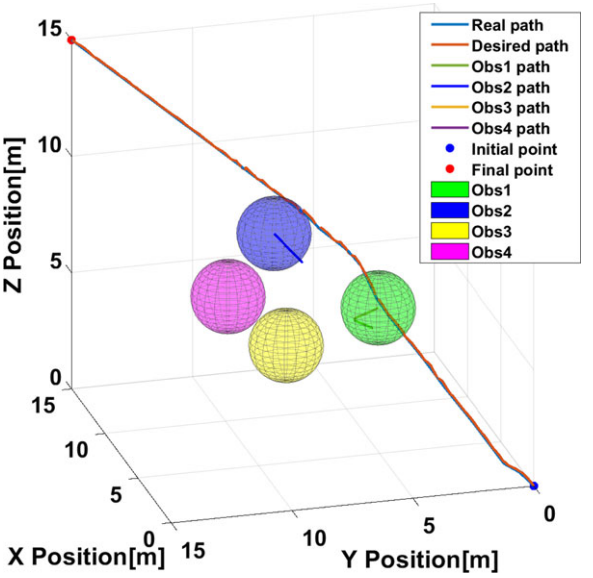


Figure 20. *Simulations: 3D path dynamics obstacles, case 2 and view 1.*

higher in the population of convergence due to the interaction with obstacle 1; then, the fitness value descends when obstacle 1 is avoided. After avoiding obstacle 1, obstacle 2 gets in the path; thus, the algorithm recalculates a new trajectory to avoid all obstacles and reach the end point. In the fitness value, Fig. 25 illustrates that after 30 seconds, the fitness value falls, which means no more obstacles get in the path and the final point. It can also be observed that between 20 and 30 seconds in the fitness value of the population of convergence, significant peaks appear due to the repopulation criterion for being close to obstacle 2. In the axes of motion, it can be seen that the change in the path is between 20 and 30 seconds, where there is a change in the X-axis and the Z-axis to be able to avoid obstacle 2.

4.3.3 Case 3

In case 3, there are three obstacles with motion and one static; obstacles 1 and 3 have circular motion, obstacle 2 has upward linear motion, and obstacle 4 has no motion. The generated 3D path is presented with two perspectives in Figs 26 and 27. It is shown that obstacle 3 prevents the path from being generated by the right; thus, the path is generated by the center; however, obstacle 1 gets into the path, and the

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

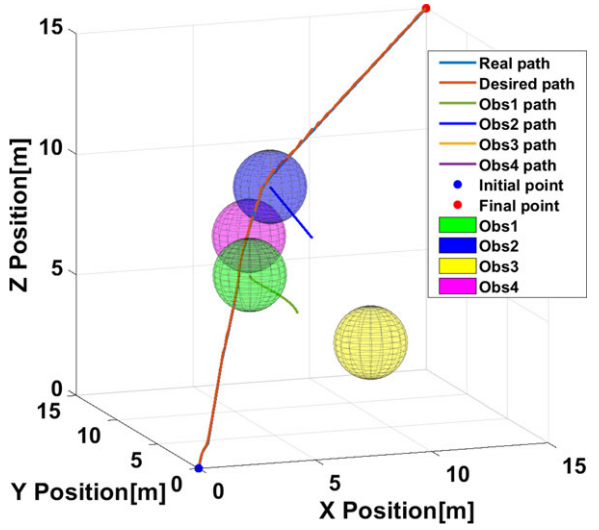


Figure 21. *Simulations: 3D path dynamics obstacles, case 2 and view 2.*

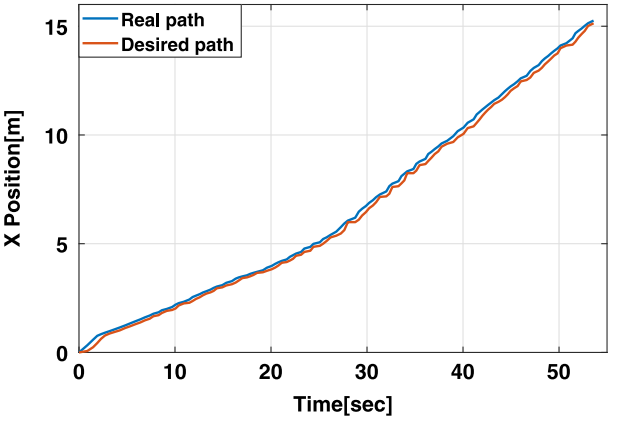


Figure 22. *Simulation: single X-axis, case 2.*

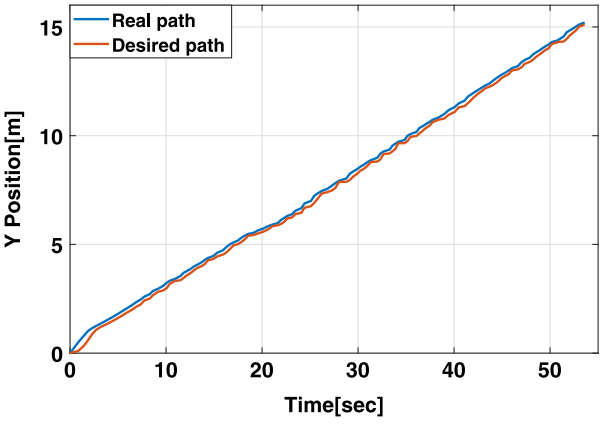


Figure 23. *Simulation: single Y-axis, case 2.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

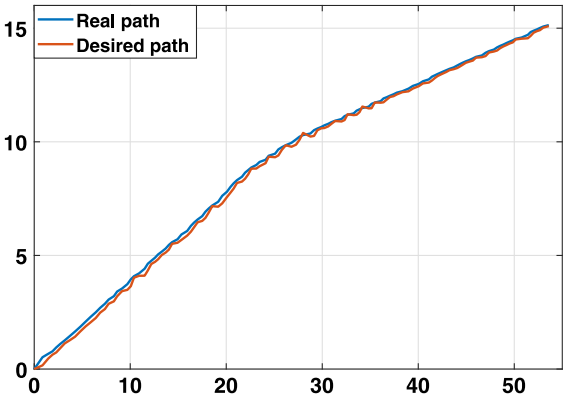


Figure 24. *Simulation: single Z-axis, case 2.*

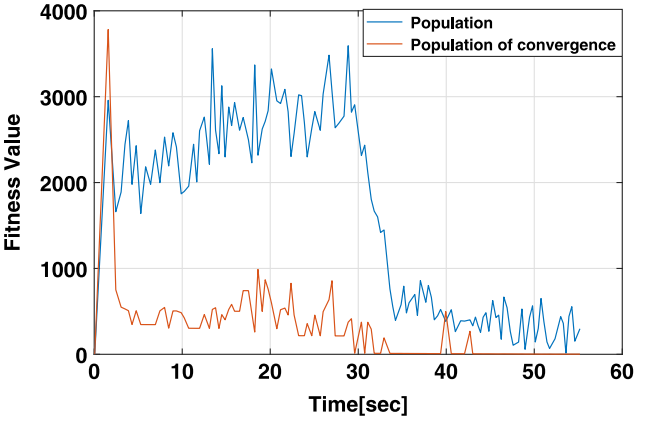


Figure 25. *Simulation: fitness value, dynamic case 2.*

change in the path to avoid the obstacle is observed around 20 seconds. This change in the path is shown in the axis motion; see Figs 28, 29 and 30. This change, in turn, is reflected in the fitness value, as shown in Fig. 31, where there is a high peak near 20 seconds; it is also seen that it affects not only the total population but also the convergence population. These peaks mean that some individuals of the population of convergence become inefficient but are immediately eliminated by genetic operators in the subsequent iterations. After avoiding obstacle 1, obstacle 2 gets in the way of the path, but obstacle 4 also gets in the way; thus, the algorithm generates the path below obstacle 2 and above obstacle 4. This change is reflected in the axes of motion Y and Z around the 30 seconds; this is also reflected in the fitness value, where there is higher. By avoiding obstacles 2 and 4, all obstacles are avoided, and the fitness value decreases until it reaches zero. These path changes generated to avoid all the obstacles are more reflected in the Z-axis motion; there are two changes in height until reaching the final point. It can be seen that there are more significant changes compared to the X-axis and Y-axis.

4.3.4 Case 4

In case 4, the four obstacles are in motion. The path generated in 3D is shown in Figs 32 and 33; it is shown that most obstacles are loaded to the right; thus, the algorithm generates the path by the left and above. In the first seconds, the algorithm generates a path that avoids all obstacles, and then obstacle 2 gets in the path; thus, the algorithm generates a change in the path to avoid the obstacles. This change in the path is reflected more significantly in the graph of the motion on the X-axis between 20 and 30

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

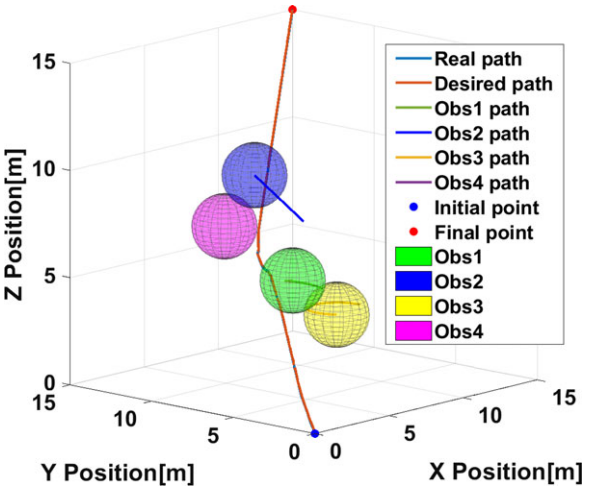


Figure 26. *Simulation: 3D path dynamics obstacles, case 3 and view 1.*

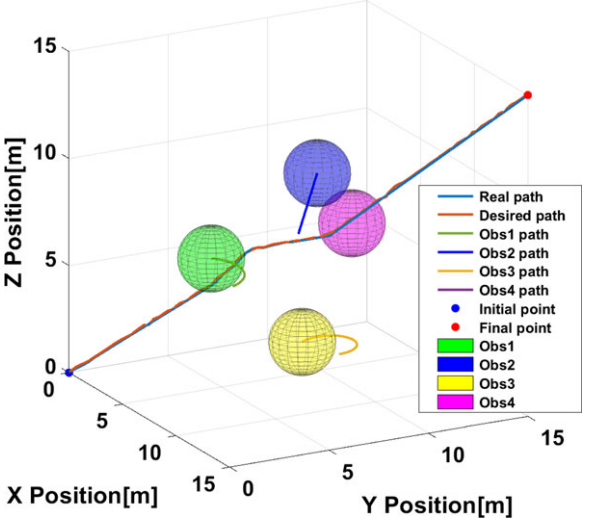


Figure 27. *Simulation: 3D path dynamics obstacles, case 3 and view 2.*

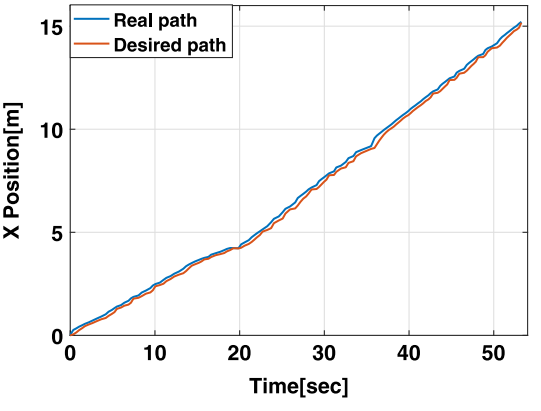


Figure 28. *Simulation: single X-axis, case 3.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

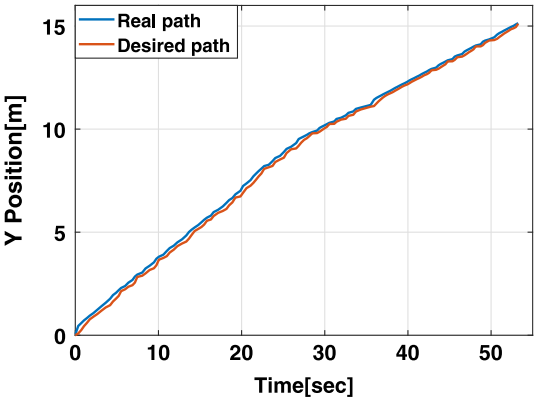


Figure 29. *Simulation: single Y-axis, case 3.*

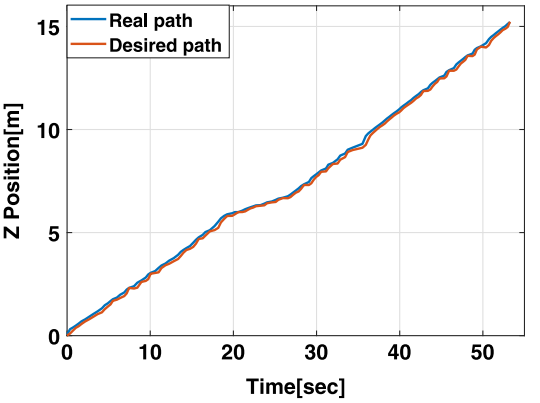


Figure 30. *Simulation: single Z-axis, case 3.*

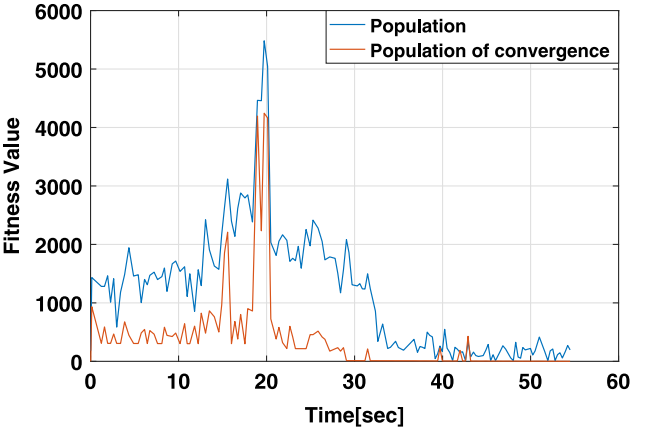


Figure 31. *Simulation: fitness value, dynamic case 3.*

seconds, as seen in Fig. 34, while Figs 35 and 36 show the *Y* and *Z* motions, respectively. In the fitness value shown in Fig. 37, there is a peak around 20 and 30 seconds, indicating that obstacle 2 suddenly gets in the path. Finally, the UAV avoids obstacle 2, and therefore, all obstacles and the fitness value fall after the 30 seconds, indicating that it finds an obstacle-free path until reaching the final point.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

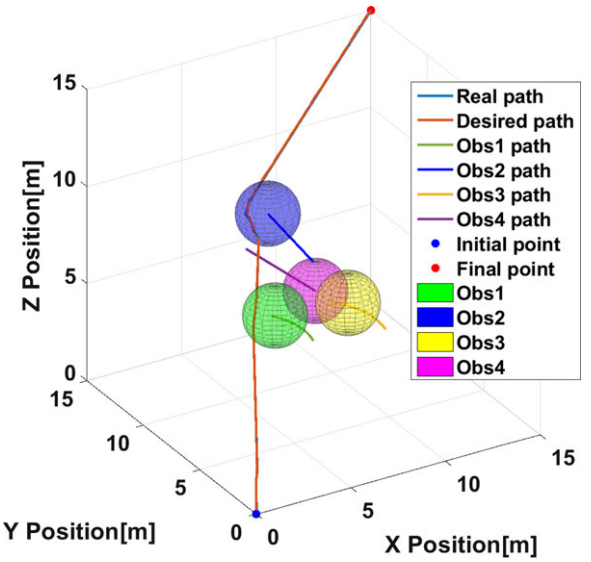


Figure 32. *Simulation: 3D path dynamics obstacles, case 4 and view 1.*

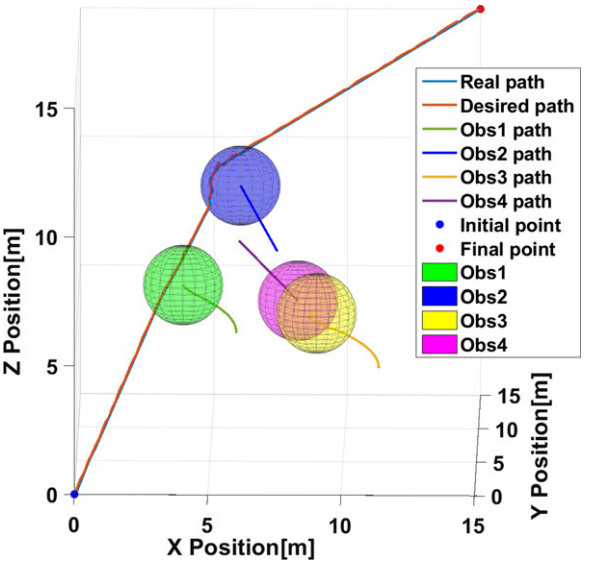


Figure 33. *Simulation: 3D path dynamics obstacles, case 4 and view 1.*

4.4 Experimental results

The experimental tests were developed in the space of 4m × 4m × 4m, with an initial point at (−2m, −2m, 0.5m), and with a final point of (1.8m, 1.8m, 1m), with a UAV constant velocity of 0.5m/s. Obstacles have a radius of 0.7m.

4.4.1 Case 1

In case 1, a case with static obstacles is presented, Figs 38 and 39 show the generated 3D path. It can be seen that the path is generated in two motions. Figures 40, 41, and 42 show the axes of motion in X,

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

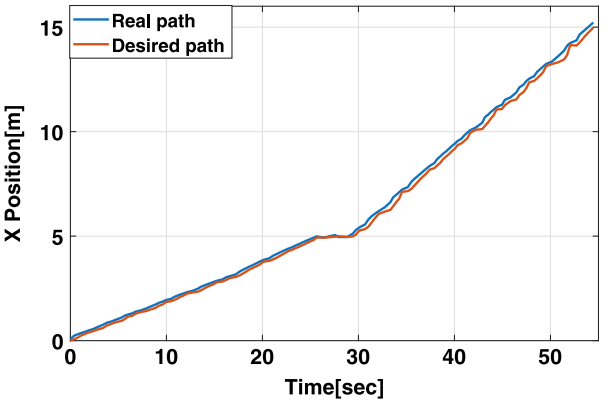


Figure 34. *Simulation: single X-axis, case 4.*

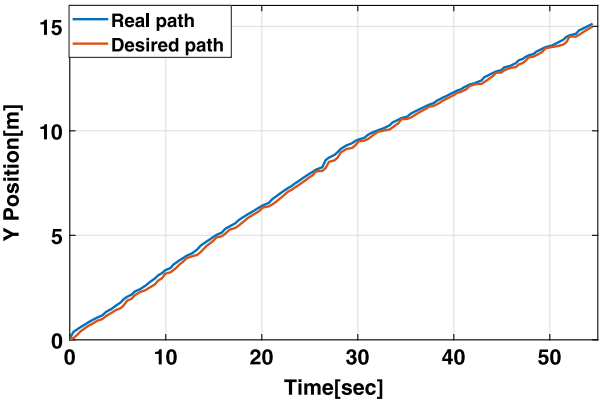


Figure 35. *Simulation: single Y-axis, case 4.*

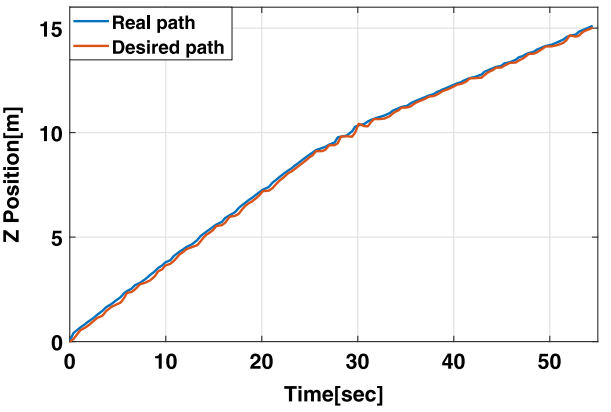


Figure 36. *Simulation: single Z-axis, case 4.*

Y and *Z*, respectively. It can be seen that the algorithm finds a path, and around 2 seconds, there is a change in the path, while the next change in the path is around 6 seconds. The fitness value, as shown in Fig. 43, confirms that around the 3 seconds the fitness value falls, but between the 4 and 6 seconds there is an increment in the fitness value. This increment means that when beginning the algorithm before

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

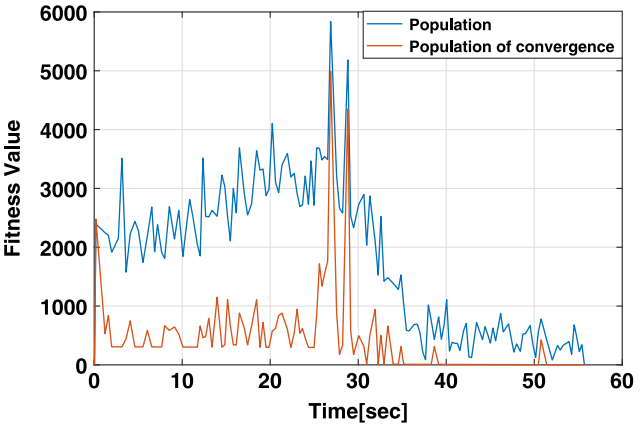


Figure 37. *Simulation: fitness value, dynamic case 4.*

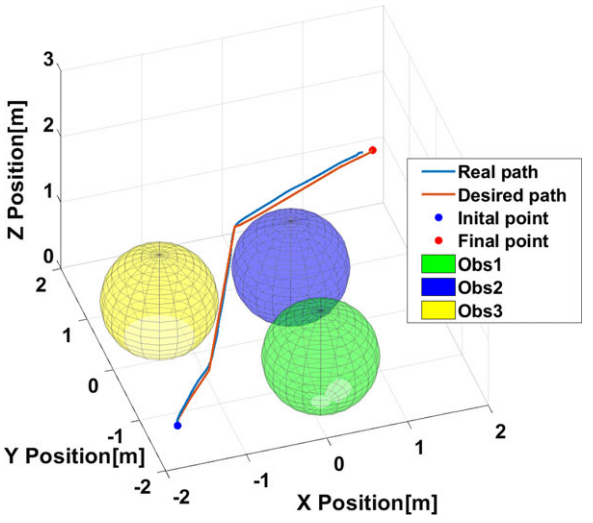


Figure 38. *Experiment: 3D path dynamic obstacles, case 1 and view 1.*

2 seconds, it finds a path that avoids all obstacles; for this reason, the fitness value decreases. After 3 seconds for a short moment, there is an increment of the fitness value; the increment of the fitness value is due to the criteria of repopulation, and this criterion is reflected when the UAV passes near the obstacles. The UAV moves until it avoids the obstacle 2 by left and above, finally it avoids all the obstacles and follows straight to the final point; this is reflected in the decline in the fitness value to zero after 8 seconds.

4.4.2 Case 2

In case 2, it can be seen in Figs 44 and 45 that obstacle 2 has two motions while obstacles 1 and 3 are static. It is shown that in the first instance, the UAV has a free path, but after 2 seconds, obstacle 2 gets in the path. Thus, a change is made in the path; this change can be seen in Figs 46, 47 and 48 which shows the motion axes, *X*, *Y* and *Z*, respectively. This change in the path, in turn, is seen in the fitness value, Fig. 46, where the fitness value of the population of convergence between 0 and 2 seconds is almost zero but then it increases when obstacle 2 gets in the path. To avoid obstacle 2, the algorithm generates the

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

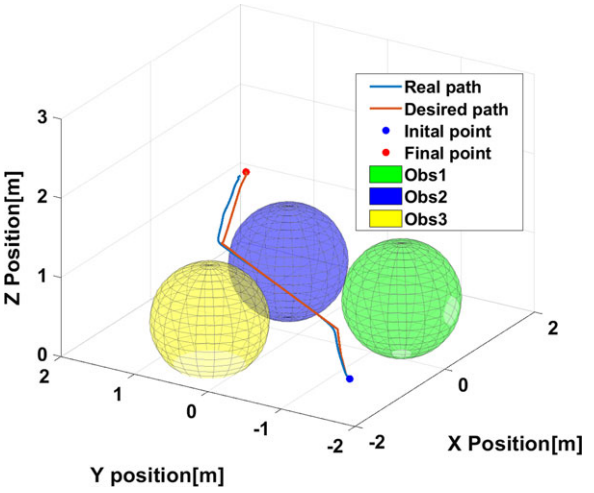


Figure 39. *Experiment: 3D path static obstacles, case 1 and view 2.*

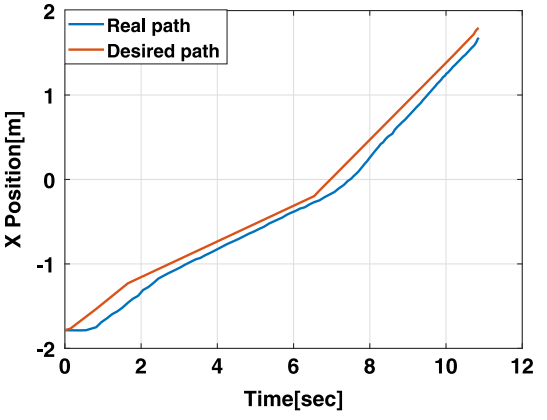


Figure 40. *Experiment: single X-axis, case 1.*

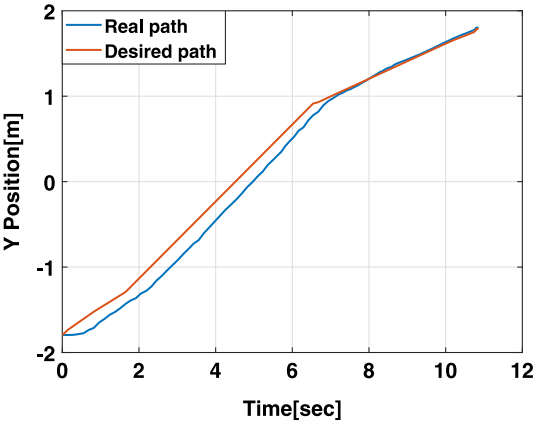


Figure 41. *Experiment: single Y-axis, case 1.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

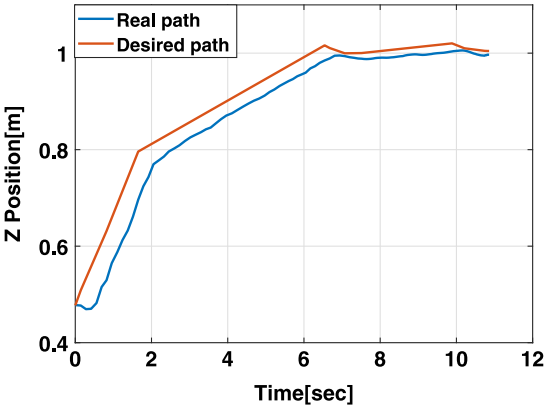


Figure 42. *Experiment: single Z-axis, case 1.*

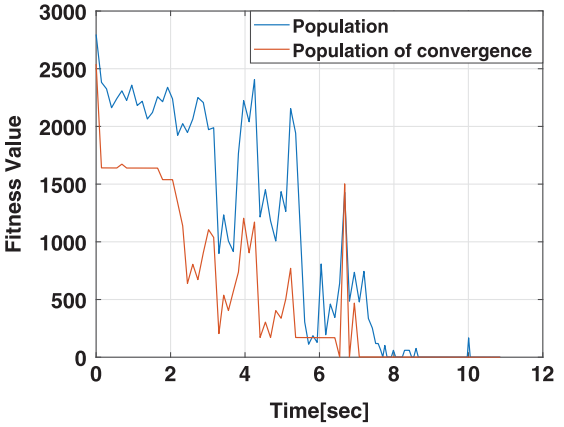


Figure 43. *Experiment: fitness value, dynamic case 1.*

path above; this is appreciated in the second view of the path, as shown in Fig. 45, and the motion in the *Z*-axis around 2 seconds, as shown in Fig. 48. However, obstacle 2 again has a second upward motion, so that the algorithm generates a path avoiding the obstacle to the right and with a downward motion. This falls in height as it is seen in the same way in the *Z*-axis graph near the 7 seconds where the 1.8m reaches the desired height of 1m. The peak in the fitness value, where the obstacle obstructs the path during the second motion, it can be observed near the 7-second mark, as shown in Fig. 49. Then, when the obstacle is avoided, the fitness value descends almost to zero after 10 seconds.

4.4.3 Case 3

In case 3, there are two obstacles in motion. It can be seen in the 3D path generated in Figs 50 and 51; it is noted that at first seconds, there is an accessible or free path, but then obstacle 2 gets in the path so that the algorithm generates a path by the left to avoid obstacle 2 after obstacle 1 gets in the path. Thus, the algorithm generates the path by the right to avoid the obstacle. Figure 51 shows that it is not only a motion in a single plane but that there are significant changes in height. Figures 52, 53 and 54 show the axes of motion, *X*, *Y* and *Z*, respectively. In the motion in the *Z*-axis near the 6 seconds, there is a descent in height, and then it rises until it reaches the desired height. The fitness value, Fig. 55, shows how, in the beginning, it finds a free path around 2 seconds when obstacle 2 gets in the path, and

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

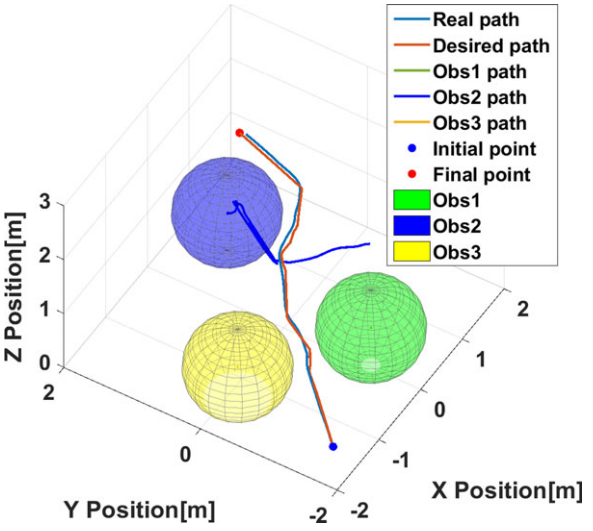


Figure 44. *Experiment: 3D path dynamics obstacles, case 2 and view 1.*

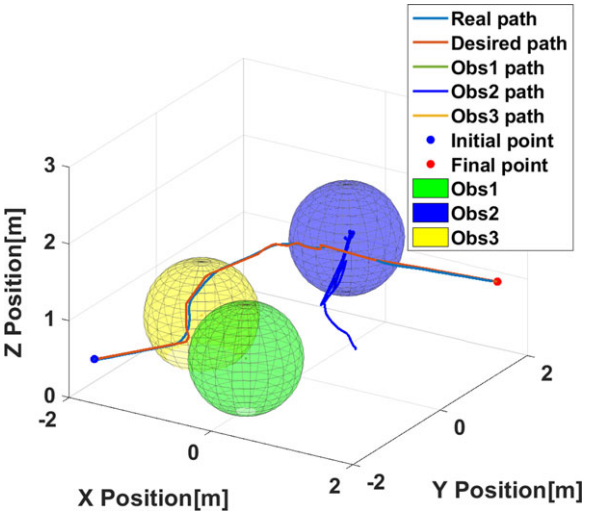


Figure 45. *Experiment: 3D path dynamics obstacles, case 2 and view 2.*

immediately the algorithm changes the path. It can be seen that obstacle 1 gets in the path, then there is a change in the path around 6 seconds; and also, there is an increase in the fitness value. Finally, the algorithm finds a free path, and the fitness value descends to zero. It can be observed that there is a peak around 10 seconds, as no waypoint was found to reach the final point before that time.

4.4.4 Case 4

In case 4, there are three obstacles in motion. Figures 56 and 57 show the generated 3D path; it can be seen how the path avoids all obstacles. Figures 58, 59 and 60 illustrate all the motions of the path. The axes of motion show a change in the path around 2 seconds; this change corresponds to obstacles 1 and 2 and how these get in the path; then it can be seen that the UAV avoided the obstacles above; this is appreciated in the path in the motion of the Z-axis. After avoiding obstacles 1 and 2, there is a small

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

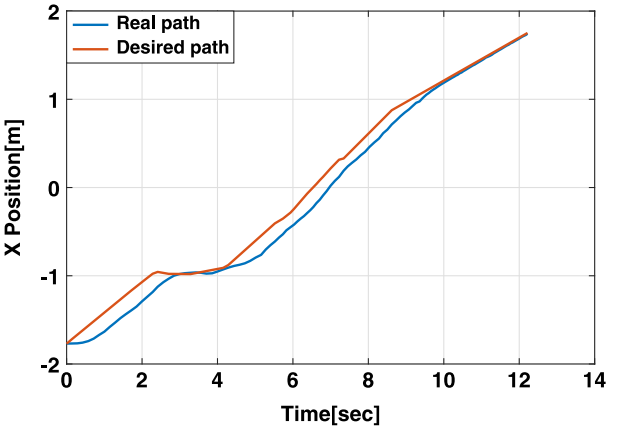


Figure 46. *Experiment: single X-axis components, case 2.*

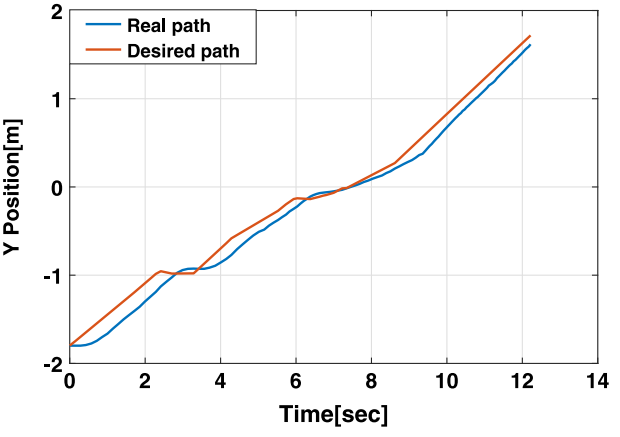


Figure 47. *Experiment: single Y-axis components, case 2.*

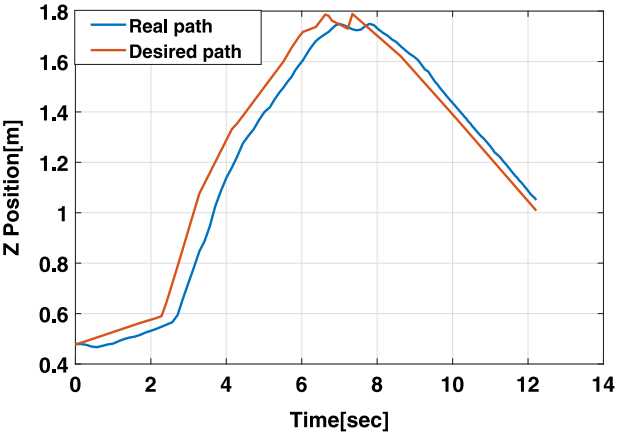


Figure 48. *Experiment: single Z-axis components, case 2.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

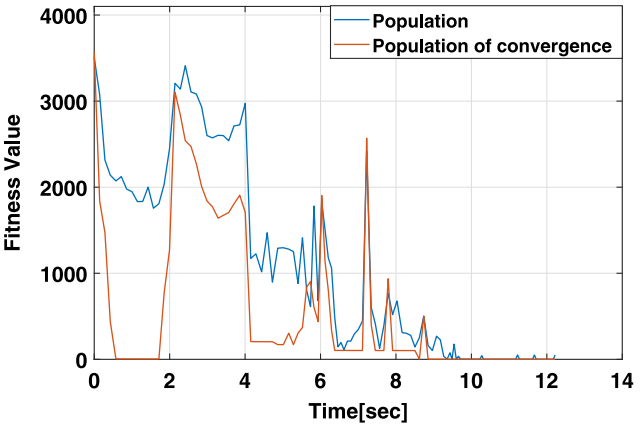


Figure 49. *Experiment: fitness value, dynamic case 2.*

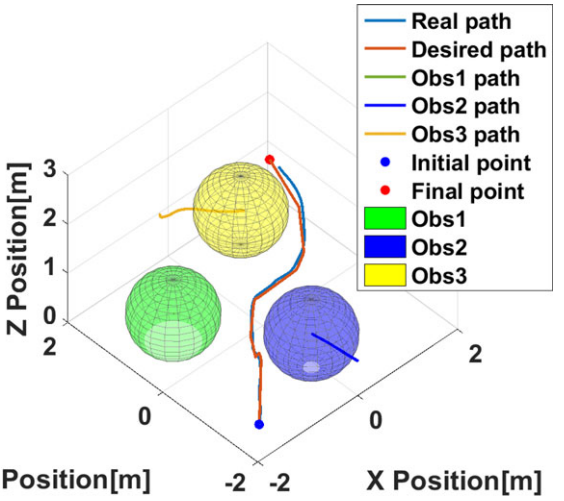


Figure 50. *Experiment: 3D path dynamics obstacles, case 3 and view 1.*

decrease in fitness value because, for a moment, there is no obstacle in the path. However, later, there is an increase in fitness value due to obstacle 3 getting in the path with a motion from the bottom to the top, which forces the UAV to avoid by moving upwards to avoid the obstacle. Around 10 seconds, it is observed that the algorithm finds a path free of obstacles; the path changes, in turn, are reflected in the fitness value, Fig. 61.

5.0 Conclusions

In this work, a genetic algorithm of path planning was proposed to navigate in static and dynamic environments with characteristics of mobile obstacle avoidance. The population of convergence criterion allows for a high mutation and, thus, diversity in the population without losing convergence. In a static environment, the comparison between the proposed algorithm against GA with low mutation and GA with high mutation but without a population of convergence has shown that the algorithm has obtained the best result on more occasions in a few iterations, demonstrating its ability to consistently find the best path. For the dynamic environment, where obstacles are in motion, the criteria of repopulation,

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

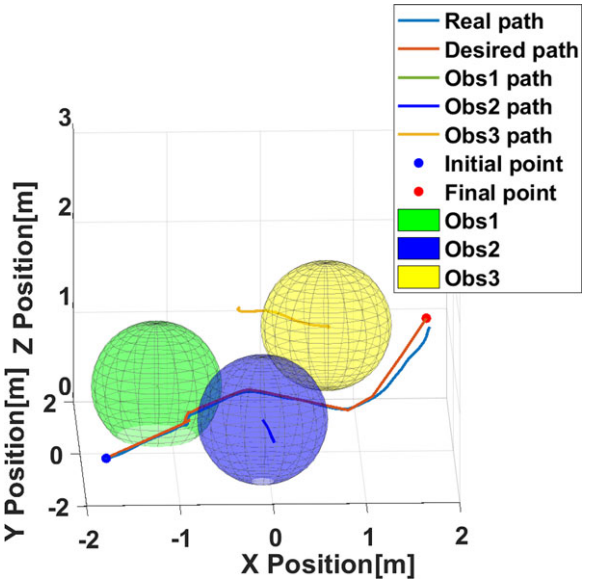


Figure 51. *Experiment: 3D path dynamics obstacles, case 3 and view 2.*

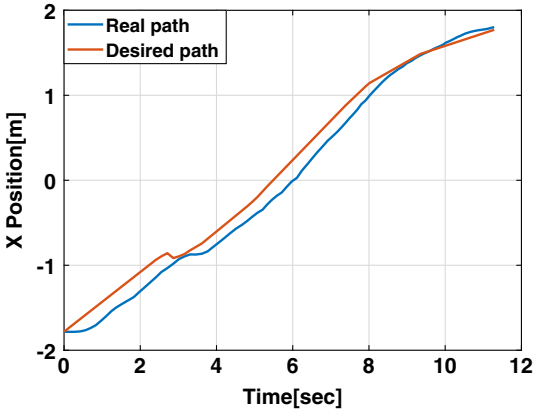


Figure 52. *Experiment: single X-axis, case 3.*

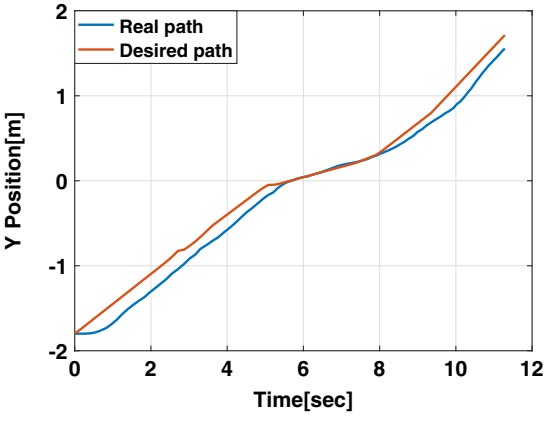


Figure 53. *Experiment: single Y-axis, case 3.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

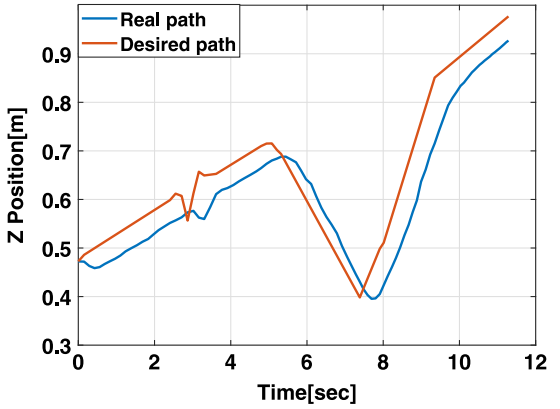


Figure 54. *Experiment: single Z-axis, case 3.*

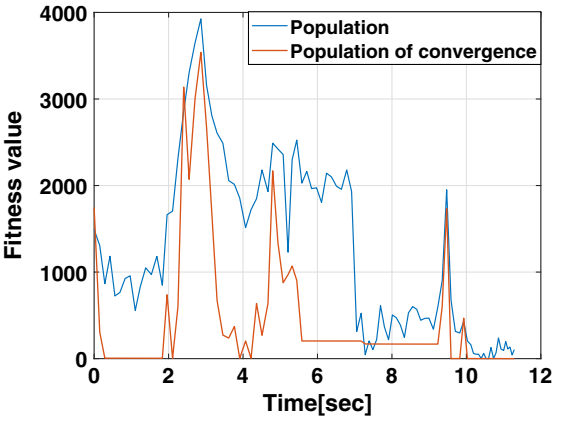


Figure 55. *Experiment: fitness value, dynamic case 3.*

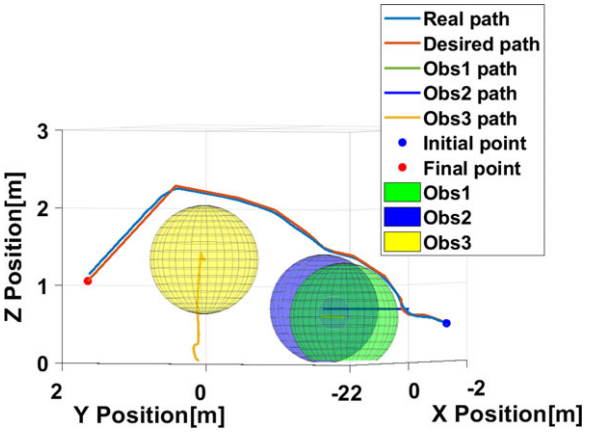


Figure 56. *Experiment: 3D path dynamics obstacles, case 4 and view 1.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

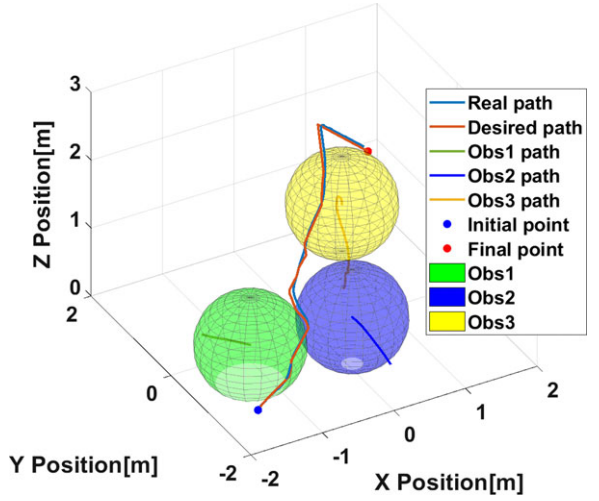


Figure 57. *Experiment: 3D path dynamics obstacles, case 4 and view 2.*

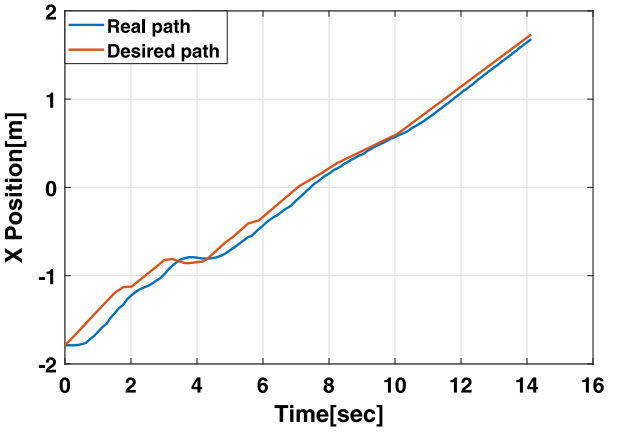


Figure 58. *Experiment: single X-axis, case 4.*

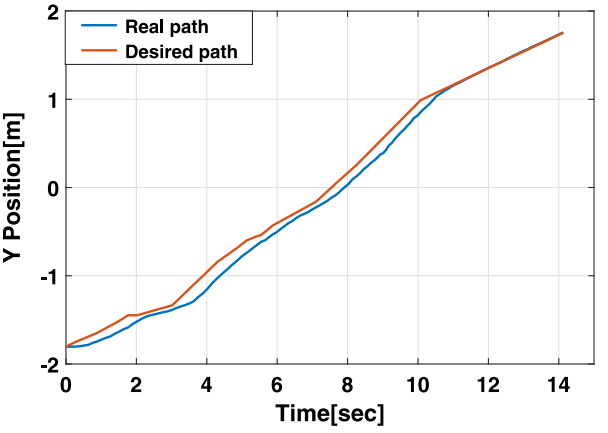


Figure 59. *Experiment: single Y-axis, case 4.*

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

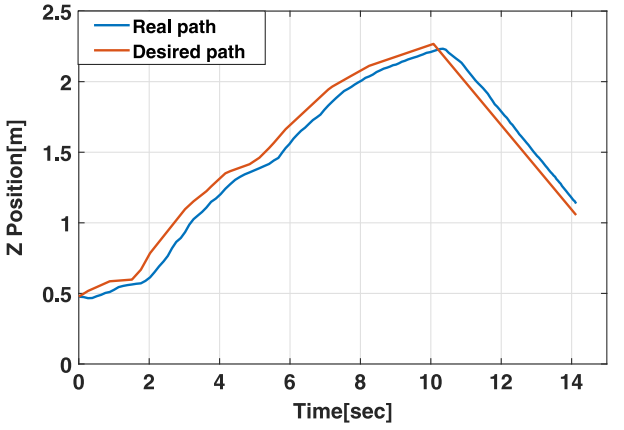


Figure 60. *Experiment: single Z-axis, case 4.*

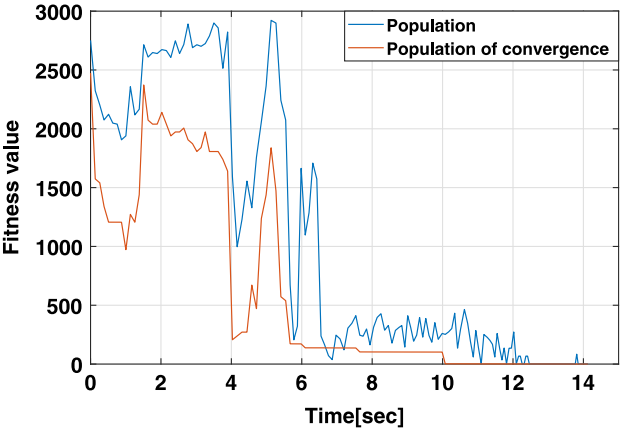


Figure 61. *Experiment: fitness value, dynamic case 4.*

high mutation, and putting the coordinates of the final point in the convergence population have shown an efficient response to the mobile obstacles and have generated an optimised path.

The identified model was used for numerical simulations to show that the algorithm can effectively perform and react to different environments. These simulations allowed the algorithm to be implemented in a larger 15m $\times$ 15m $\times$ 15m environment across four test cases. The fitness value graphs, which indicated path performance, showed a decreasing pattern, suggesting continuous improvement in the path, almost reaching zero. Additionally, the fitness graphs showed sudden peaks when obstacles obstructed the path, but these quickly declined, demonstrating the algorithm ability to find the best path.

The experimental tests were conducted in a laboratory setting within an area of 4m $\times$ 4m $\times$ 4m using Crazyflie quadrotors, demonstrating the successful implementation of genetic algorithms for online path planning. In these experimental tests, a low computational cost was maintained, making the selection of criteria highly relevant, allowing the genetic algorithm to efficiently find paths. The use of small populations reduced the computational costs, while a high mutation rate ensured that the waypoint did not stagnate. Placing the final point in the convergence population allowed the UAV to fly directly to the destination when it found an obstacle-free path. In addition, repopulation provided path options when the UAV was near the waypoint. As in the simulations, the fitness value graphs mostly showed a decreasing pattern, indicating continuous improvement in the path. However, these paths exhibited more sudden peaks, although the path graphs showed no collisions, suggesting that greater diversity in the population was needed to find paths that avoided all obstacles.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

Acknowledgments. This research work is supported by the Office of Naval Research Global through the grant number N62909-20-1-2030.

Competing interests. The authors declare that there is no competing interests.

References

[1] Majeed, A. and Lee, S. A fast global flight path planning algorithm based on space circumscription and sparse visibility graph for unmanned aerial vehicle, *Electronics*, 2018, **7**, (12), p 375.
[2] Wang, H., Lyu, W., Yao, P., Liang, X. and Liu, C. Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system, *Chin. J. Aeronaut.*, 2015, **28**, pp 229–239.
[3] Aggarwal, S. and Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, *Comput. Commun.*, 2020, **149**, pp 270–299.
[4] Yingkun, Z. Flight path planning of agriculture UAV based on improved artificial potential field method, Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018, 2018.
[5] Dimas Flores, G.E., Espinoza Quesada, E.S., Salazar Cruz, S., Garcia Carrillo, L.R. and Lozano, R. Online UAS local path planning algorithm for outdoors obstacle avoidance based on attractive and repulsive potential fields, 2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016, USA, 2016.
[6] Samaniego, F., Sanchis, J., Garcia-Nieto, S. and Simarro, R. UAV motion planning and obstacle avoidance based on adaptive 3D cell decomposition: Continuous space vs discrete space, 2017 IEEE 2nd Ecuador Technical Chapters Meeting, ETCM 2017, January 2017.
[7] Blasi, L., D'Amato, E., Mattei, M. and Notaro, I. Path planning and real-time collision avoidance based on the essential visibility graph, *Appl. Sci.*, 2020, **10**, (16), p 5613.
[8] Xue, Y. and Sun, J.Q. Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm, *Appl. Sci.*, 2018, **8**, (9), p 1425.
[9] Aghda, S.A.F. and Mirfakhraei, M. Improved routing in dynamic environments with moving obstacles using a hybrid fuzzy-genetic algorithm, *Future Gener. Comput. Syst.*, 2020, **112**, pp 250–257.
[10] Shao, S., Peng, Y., He, C. and Du, Y. Efficient path planning for UAV formation via comprehensively improved particle swarm optimization, *ISA Trans.*, 2020, **97**, pp 415–430.
[11] Shiri, H., Park, J. and Bennis, M. Massive autonomous UAV path planning: A neural network based mean-field game theoretic approach, 2019 IEEE Global Communications Conference, GLOBECOM 2019, December 2019.
[12] Konatowski, S. and Pawlowski, P. Ant colony optimization algorithm for UAV path planning, 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018, April 2018.
[13] Patle, B.K., Parhi, D.R., Jagadeesh, A. and Kashyap, S.K. Matrix-binary codes based genetic algorithm for path planning of mobile robot, *Comput. Electr. Eng.*, 2018, **67**, pp 708–728.
[14] Atyabi, A. and Powers D. Review of classical and heuristic-based navigation and path planning approaches, *Int. J. Adv. Comput. Technol. IJACT,* 2013, **5**, pp 1–14.
[15] Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press, USA, 1992.
[16] Lamini, C., Benhlima, S. and Elbekri, A. Genetic algorithm based approach for autonomous mobile robot path planning, *Procedia Comput. Sci.*, 2018, **127**, pp 180–189.
[17] Wang, Y. and Chen, W. Path planning and obstacle avoidance of unmanned aerial vehicle based on improved genetic algorithms, Proceedings of the 33rd Chinese Control Conference, CCC 2014, 2014.
[18] Jayaweera, H.M. and Hanoun, S. A dynamic artificial potential field (D-APF) UAV path planning technique for following ground moving targets, *IEEE Access*, 2020, **8**, pp 192760–192776.
[19] Pan, Z., Zhang, C., Xia, Y., Xiong, H. and Shao, X. An improved artificial potential field method for path planning and formation control of the multi-UAV systems, *IEEE Trans. Circ. Syst. II Exp. Briefs*, 2022, **69**, (3), pp 1129–1133.
[20] Elhoseny, M., Shehab, A. and Yuan, X. Optimizing robot path in dynamic environments using genetic algorithm and bezier curve, *J. Intell. Fuzzy Syst.*, 2017, **33**, pp 2305–2316.
[21] Roberge, V., Tarbouchi, M. and Labonte, G. Fast genetic algorithm path planner for fixed-wing military UAV using GPU, *IEEE Trans. Aerospace and Electron. Syst.*, 2018, **54**, pp 2105–2117.
[22] Elhoseny, M., Tharwat, A. and Hassanien, A.E. Bezier curve based path planning in a dynamic field using modified genetic algorithm, *J. Comput. Sci.*, 2018, **25**, pp 339–335.
[23] Shivgan, R. and Dong, Z. Energy-efficient drone coverage path planning using genetic algorithm, IEEE International Conference on High Performance Switching and Routing, May 2020.
[24] Arantes, M.S., Arantes, J.S., Toledo, C.F.M. and Williams, B.C. A hybrid multi-population genetic algorithm for UAV path planning, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2016, 2016, pp 853–860.
[25] Pehlivanoglu, Y.V. and Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems, *Appl. Soft Comput.*, 2021, **112**, p 107796.
[26] Pehlivanoglu, Y.V. A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous UAV, *Aerospace Sci. Technol.*, 2012, **16,** (1), pp 47–55.
[27] Zhang, C., Zhou, W., Qin, W. and Tang, W. A novel UAV path planning approach: Heuristic crossing search and rescue optimization algorithm, *Exp. Syst. Appl.*, 2023, **215**, p 119243.

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press

[28] Pan, Y., Yang, Y. and Li, W. A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-UAV, *IEEE Access*, 2021, **9**, pp 7994–8005.
[29] Chen, B., Lai, S., Chen, C., Shu, P., Chen, S., Lai, Z. and Xu, L. UAV path planning based on improved genetic algorithm, 3rd International Symposium on Robotics & Intelligent Manufacturing Technology ISRIMT 2021, 2021, pp 229–232.
[30] Bitcraze. Crazyflie 2.1, https://www.bitcraze.io/products/crazyflie-2-1/,2011
[31] Giernacki, W., Skwierczynski, M., Witwicki, W., Wronski, P. and Kozierski, P. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering, 22nd International Conference on Methods and Models in Automation and Robotics, MMAR 2017, 2017, pp 37–42.
[32] Preiss, J.A., Honig, W., Sukhatme, G.S. and Ayanian, N. Crazyswarm: A large nano-quadcopter swarm, IEEE International Conference on Robotics and Automation ICRA 2017, 2017, pp 3299–3304.

Cite this article: Gutierrez-Martinez M.A., Rojo-Rodriguez E.G., Cabriales-Ramirez L.E., Estabridis K. and Garcia-Salazar O. (2025). Genetic algorithm-based path planning of quadrotor UAVs on a 3D environment. *The Aeronautical Journal*, **129**, 902–938. https://doi.org/10.1017/aer.2024.132

https://doi.org/10.1017/aer.2024.132 Published online by Cambridge University Press