drones

MDPI

Article

An Integrated Geometric Obstacle Avoidance and Genetic Algorithm TSP Model for UAV Path Planning

Dipraj Debnath [1,2,*], Fernando Vanegas [1,2], Sebastien Boiteau [1,2] and Felipe Gonzalez [1,2]

[1] School of Electrical Engineering and Robotics, Queensland University of Technology (QUT), 2 George Street, Brisbane, QLD 4000, Australia; f.vanegasalvarez@qut.edu.au (F.V.); sebastienthierry.boiteau@hdr.qut.edu.au (S.B.); felipe.gonzalez@qut.edu.au (F.G.)
[2] QUT Centre of Robotics (QCR), Queensland University of Technology, Level 11, QUT S Block, Brisbane, QLD 4000, Australia
* Correspondence: dipraj.debnath@hdr.qut.edu.au; Tel.: +61-0493610399

Abstract: In this paper, we propose an innovative approach for the path planning of Uninhabited Aerial Vehicles (UAVs) that combines an advanced Genetic Algorithm (GA) for optimising missions in advance and a geometrically based obstacle avoidance algorithm (QuickNav) for avoiding obstacles along the optimised path. The proposed approach addresses the key problem of determining an optimised trajectory for UAVs that covers multiple waypoints by enabling efficient obstacle avoidance, thus improving operational safety and efficiency. The study highlights the numerous challenges for UAV path planning by focusing on the importance of both global and local path planning approaches. To find the optimal routes, the GA utilises multiple methods of selection to optimise trajectories using the Cartesian Coordinate System (CCS) data transformed from a motion capture system. The QuickNav algorithm applies linear equations and geometric methods to detect obstacles, guaranteeing the safe navigation of UAVs and preventing real-time collisions. The proposed methodology has been proven useful in reducing the total distance travelled and computing times and successfully navigating UAVs across different scenarios with varying numbers of waypoints and obstacles, as demonstrated by simulations and real-world UAV flights. This comprehensive approach provides advantageous perspectives for real-world applications in a variety of operational situations and improves UAV autonomy, safety, and efficiency.

Keywords: UAV path planning; obstacle avoidance; travelling salesmen problem; genetic algorithm; geometrical-based approach; optimisation

Citation: Debnath, D.; Vanegas, F.; Boiteau, S.; Gonzalez, F. An Integrated Geometric Obstacle Avoidance and Genetic Algorithm TSP Model for UAV Path Planning. *Drones* **2024**, *8*, 302. https://doi.org/10.3390/drones8070302

Academic Editors: Kaiyu Qin, Haitao Nie, Yikang Yang, Xue Li, Jinliang Shao, Lei Shi and Mengji Shi

Received: 15 May 2024
Revised: 21 June 2024
Accepted: 3 July 2024
Published: 7 July 2024

1. Introduction

Modern technological developments have made UAVs a part of contemporary human culture. UAV technology has developed significantly in recent decades and is currently utilised extensively in military and civilian applications. UAV autonomous flight systems with obstacle avoidance received attention from scholars worldwide, becoming a topic of research in UAVs [1]. The future evolution of UAVs will inevitably include autonomous mission performance and one of the most essential developments to enhance UAV autonomy and guarantee flight safety is path planning [2,3]. The successful deployment of UAVs at their designated mission site relies on effective path planning, which serves as the fundamental aspect of the flight towards the mission location [4].

Several sectors, including the military, farming, industry, and academics, have long been interested in UAV route planning [5–7]. Path planning is an essential method for UAVs, encompassing two main components: global path planning (offline) and local path planning (online) [8–10]. Global planning, also known as static path planning, involves determining the path when complete environmental information is available. In circumstances in which the environment is uncertain or just partially understood, local planning methods are

used for path planning. Local planning can be considered as a form of dynamic path planning [11,12].

The challenges of path planning involve determining the optimal sequence of valid routes for travel to achieve strategic objectives while reducing expenses, such as the total length of the trip. This technique has traditionally been associated with addressing the Travelling Salesmen Problem (TSP), in which the cost of travel is determined by the distance between consecutive sites on the road [13]. Although UAV routes are significantly more complex, such an approach supports generic optimisation, which shares a similar difficulty with UAV path planning [14].

Moreover, UAVs are progressively being employed for gathering data in inaccessible regions through the use of Wireless Sensor Networks (WSNs) [15,16]. In these situations, UAVs are required to travel to specified waypoints in order to acquire data. This represents a challenge for solving the TSP as it involves optimising journey distance and minimising operational time [17,18]. Furthermore, in precision agriculture, it is important to have effective path planning for UAVs to navigate specific locations that have been identified by images or data analysis, especially for tasks like targeted pesticide application for controlling weeds [19]. Aligning the optimisation of these paths of flight with the aims of TSP optimisation requires ensuring complete coverage while minimising the expenditure of time [20].

Numerous studies have been conducted on path planning and obstacle avoidance algorithms, leading to the development of different study methodologies. The path planning algorithms can be categorised into two groups: classic optimisation methods and new intelligence algorithms. Conventional optimisation algorithms contain the steepest descent approach, optimal control technique, and derivative connection approach. The most prevalent instances of modern intelligence algorithms are Genetic Algorithms, Artificial Neural Networks, and Ant Colony Optimisation (ACO) [21,22].

Several conventional optimisation algorithms possess notable limitations, including inadequate adaptability, stability, and a tendency to become trapped in local optimisation, rendering them unsuitable for UAV path planning. On the other hand, the GA promotes the principles of survival and competition, subsequently selecting the most favourable outcome through the inheritance execution, continuously converging towards the optimal solution [23]. The study entitled "Comparison of Parallel Genetic Algorithm and Particle Swarm Optimisation for Real-Time UAV Path Planning" examines and establishes that the GA outperforms the Particle Swarm Algorithm in trajectory planning [24]. Nevertheless, the GA discussed in this paper has inadequate optimisation accuracy and is susceptible to local optimisation, rendering it inefficient for solving current path planning problems. On the other hand, the process of ACO necessitates a substantial quantity of information to identify the most efficient solution. Its iteration speed is inefficient, leading to a significantly greater amount of simulation time and rendering it inappropriate for the path planning problem [25,26]. Yang et al. proposed an alternative method for path optimisation combining a basic rapidly exploring random tree (RRT) algorithm and ACO [27]. However, the level of unpredictability in the RRT method is excessive, making it difficult to achieve an effective solution. Lin et al. presented a map-based offline path planning technique to construct an initial path and trajectory waypoints for fight guiding [28]. To perceive the environment, an onboard camera captures monocular images during navigation. A collision avoidance system is proposed, employing vision-based obstacle identification using optical flow. However, they achieve low success rates for multiple obstacles due to the optical flow field's inability to effectively distinguish between objects and the surroundings. Pehlivanoglu et al. introduced artificial intelligence methods combined with a GA, ACO, Voronoi diagram, and clustering approaches to tackle the path planning problem of autonomous UAV in target coverage challenges [29]. Regardless, the presence of altitude restrictions can make the resulting trajectories hazardous and therefore devoid of significance, as they may lead to collisions with the terrain surface. Hu et al. used deep reinforcement learning and the Proximal Policy Optimisation (PPO) method to improve the navigation of autonomous Unmanned Aerial Systems (UASs) in continuous state and action

spaces [30]. The limitation of this method is its inability to model intruders as agents or incorporate teamwork across multiple aircraft. Furthermore, it failed to incorporate learning difficulties related to both stationary and moving obstacles through the combination of reward functions. In another paper, AlRaslan et al. proposed the Parallel Genetic Algorithm (PGA) models and explained the way the recommended path planner can be used in parallel on multi-core processors with OpenMP [31]. However, as the number of threads increased, the parallel implementation failed to accelerate at a sustainable rate. Several hybrid intelligence algorithms that combine various methods may exhibit superior performance. However, the algorithms do not take into account the influence of obstacles on the TSP [32,33].

Heuristic search algorithms provide appealing features for UAV path planning, particularly in unpredictable circumstances because of their efficient exploration and flexibility [34]. Nevertheless, their dependence on well-crafted rules and the computational expense linked to complex surroundings can result in less-than-ideal solutions or limit real-time efficiency, especially for those solving the TSP [35]. This research acknowledges the limitations of heuristic search algorithms like A* in addressing the minimal time TSP issue, despite their effectiveness in changing circumstances with real-time modifications [36]. Although A* effectively navigates across the search area and adjusts to dynamic circumstances, its reliance on a carefully designed heuristic function might result in less-than-optimal solutions, particularly in sophisticated situations with an extensive number of waypoints [37]. Moreover, the expense of expanding the search tree might be substantial for TSP examples on a large scale, impeding real-time performance in time-sensitive situations [38].

This study introduces a unique method that combines an enhanced Genetic Algorithm (GA) with a geometrically based obstacle avoidance algorithm titled QuickNav in order to address UAV path planning with obstacle avoidance. Although previous research has examined this topic using different optimisation strategies, this approach offers a new and innovative solution. This integration aims to overcome the obstacles that have been identified in prior solutions. In this section, we delineate the key contributions of this research:

- Hybrid Optimisation for Path Planning Efficiency: This study presents a new hybrid strategy that combines the advantages of Genetic Algorithms (GAs) and geometric-based methodologies. Genetic Algorithms (GAs) are highly proficient in swiftly navigating through the search space to determine the most effective paths. On the other hand, the QuickNav algorithm guarantees obstacle avoidance that prioritises safety by utilising the CCS data transformed from a motion capture system. This integrated methodology seeks to overcome the drawbacks of exclusively depending on GAs, such as their vulnerability to getting stuck in local optimal solutions and the high computational cost of some linear path planning techniques.
- Decreased Path Planning Time for Obstacle Avoidance: The integration of QuickNav significantly improves the efficiency of computing for planning routes throughout obstacle avoidance, utilising an effective approach for faster computation times. The enhanced efficiency is particularly crucial for urgent emergency operations that necessitate rapid path optimisation with minimal delay.
- Validation Across Diverse Scenarios: The proposed GA-QuickNav combination has been tested extensively to prove its reliability and effectiveness in various scenarios. This has been accomplished by conducting simulations in environments with variable obstacle densities. The method has been evaluated in scenarios with varying numbers and configurations of obstacles to determine its effectiveness in different obstacle environments. Furthermore, real UAV flight data have been included to assess the approach's practicality and verify its performance in real-life scenarios.

2. Materials and Methods

This study employs an improved GA to solve the path planning problem in a TSP model and a geometric-based algorithm (QuickNav) is used for both detection and avoidance to find an optimised path for a specific set of waypoints in a two-dimensional envi-

ronment. The GA generates the optimal linear trajectory by utilising specified locations obtained from a motion capture system (optitrack) data in the latitude–longitude–altitude form. For indoor experiment purposes, in the Cartesian Coordinate System (CCS), these three parameters can be denoted as x, y, and z alternatively. The flight controller is responsible for executing motion control and navigation operations. Figure 1 represents the flight situation for TSP path planning. The figure depicts a possible version of the TSP specifically designed for UAVs. The map displays 13 waypoints labelled as "Waypoint". The UAV is required to visit these waypoints in the most efficient sequence. The objective is to identify the most efficient path that covers all places and returns to the initial location, minimising flight distance and avoiding obstacles between waypoints.

Figure 1. TSP path planning problem scenario for UAVs.

The primary purpose of the path planning module is to generate a sequence of predetermined optimised waypoints in the shortest amount of time for the flight controller to navigate. Since the experimentation method selected is based on an indoor GPS-denied flight area with a cartesian localisation motion capture system, we express the TSP problem using the CCS data transformed from the motion capture system coordinates. Parameter z, in this case, denotes the vertical dimension, specifically the height or altitude, which holds significant importance in the process of 3D path planning. Nevertheless, this study specifically examines the process of planning a two-dimensional course and operates under the assumption that the UAV maintains a constant altitude. This assumption is especially advantageous for the specific applications of precision agriculture and Wireless Sensor Network monitoring. It ensures that the sensors can operate consistently and provide coverage by maintaining a constant altitude. Therefore, the calculation of the z component will be excluded. The route cost will be estimated by two variables: the distance that must be travelled and the travel duration. Initially, the GA is introduced as an enhanced algorithm for identifying the most efficient route. Subsequently, the geometrical-based algorithm

(QuickNav) verifies this optimal route and checks every path for obstacle detection and avoidance. Finally, if any obstacles are encountered during the journey, the QuickNav algorithm navigates around them using the shortest feasible route. The complete process is illustrated in Figure 2.

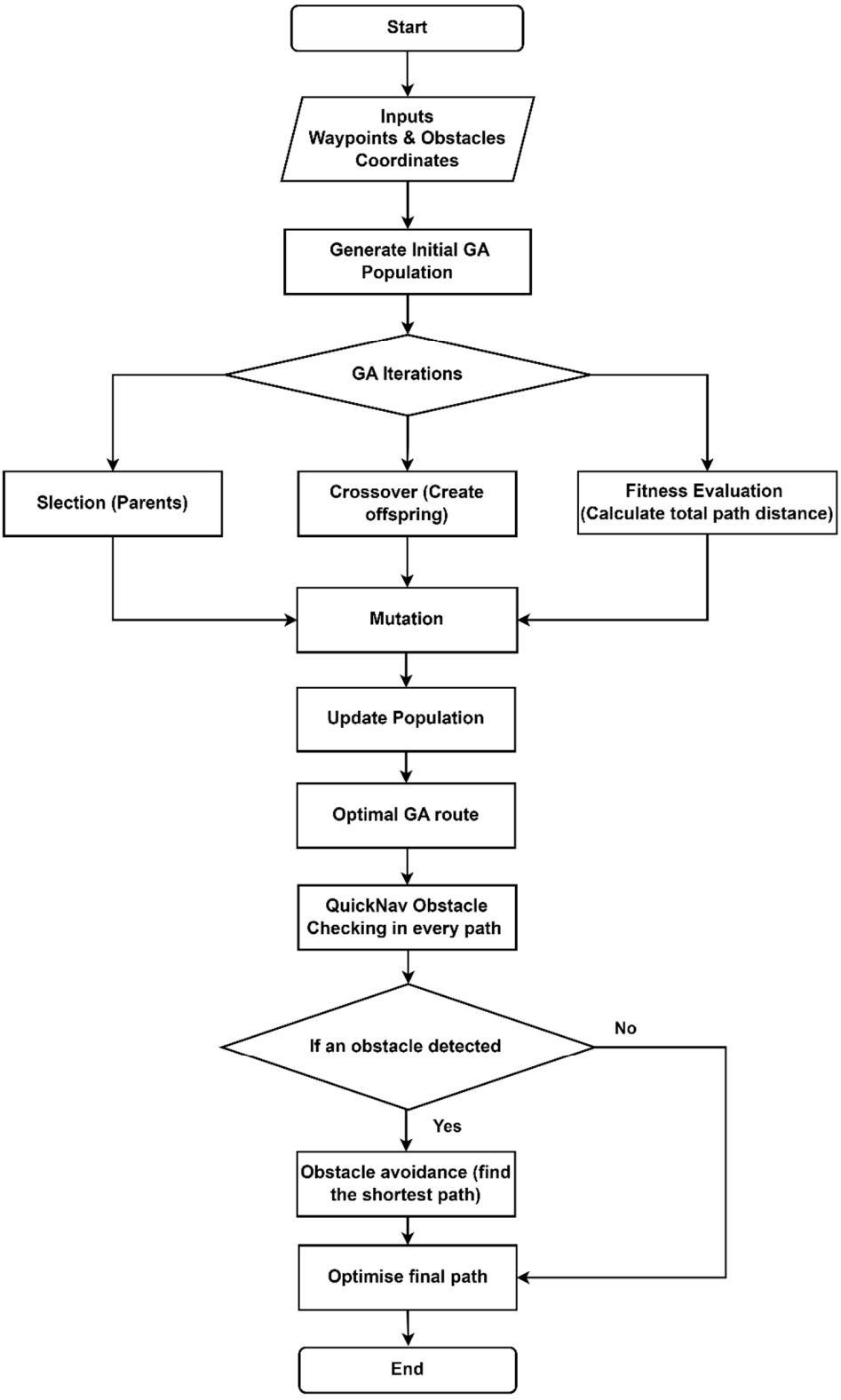

Figure 2. UAV path planning using GA and QuickNav.

In our improved GA, xy represents the matrix that contains the 2D coordinates of waypoints that we get from the motion capture system used as CGS form and d_{ij} represents the matrix that contains the distances calculated using the Euclidean distance formula:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

In this context, d_{ij} represents the Euclidean distance between waypoints i and j in a two-dimensional environment. The GA is set up with a population of potential routes, where each route is represented as a permutation of waypoints. Here, the UAV used a Genetic Algorithm as a mechanism for the selection of the route outcome. The goal is to arrive at a conclusion that is either the same as or comparable to the solution of a problem. The present layout is built upon previous model solutions. The main method used in Genetic Algorithms for developing and enhancing solutions/offspring involves the cooperation of parent chromosomes. The crossover function configuration has a greater impact on the Genetic Algorithm's outputs. First, in our advanced Genetic Algorithm, the UAV generates a population using the data from the TSP model obtained using the motion capture system shown in Figure 2. For instance, if the TSP model consists of six cities represented by the sequence [1 2 3 4 5 6], the algorithm generates the population by randomly permuting the sequence, resulting in population 1 (2 3 4 1 6 5). Subsequently, the algorithm computed the aggregate distance across all populations. Additionally, the duration of the iteration for resolving the problem in this algorithm was specified. In the population, the GA chooses twelve parents that are closest to a minimum distance and uses them to produce the next generation of offspring. The mutation process begins in the algorithm by employing sliding, swapping, and flipping operations across the cities of parent chromosomes until the iteration phase is finished. The crucial variables consist of the population size (Pop Size) and the maximum number of iterations (numIter). In this algorithm, the population size is assigned the number 12, and the iteration period for the algorithm is 1,000,000. The selection of a population size of 12 enables a sufficiently diverse genetic pool, allowing the searching of a wide range of possibilities while maintaining computational efficiency. This specific size is highly efficient in producing numerous offspring, allowing the algorithm to thoroughly explore different combinations of paths and significantly increasing the chances of avoiding suboptimal solutions. Moreover, by limiting the number of iterations to one million, the algorithm is able to thoroughly explore the solution space in a reasonable amount of time, which is essential for applications that involve efficient path planning. The large iteration count is essential for thoroughly refining the solution space, particularly when the dataset size expands and the complexity of the path planning challenge increases. The parameter configuration was selected after extensive testing, which demonstrated its ability to effectively generate optimal pathways under different conditions. This configuration is well-suited for rapid deployment in UAV applications, corresponding with operational requirements. Although these parameters were successful in our experiments, they can be modified to meet the specific needs of the operational environment, ensuring that the Genetic Algorithm remains flexible and adaptable to numerous conditions. The iterative GA procedure entails the evaluation of routes using a fitness function.

$$f(R) = \sum_{k=1}^{n} d_{R_k R_{k+1}} \quad (2)$$

This function computes the cumulative distance of a given route R by considering the lengths between each succeeding waypoint. The population undergoes iterative application of selection, crossover, and mutation procedures, which introduce genetic variation.

The algorithm's convergence is monitored by tracking the minimum attainable total distance (*minDist*).

$$minDist = \min\left\{\sum_{k=1}^{n} d_{R_k R_{k+1}}\right\} \quad (3)$$

In this context, the representation of R indicates a specific route, while $d_{RkR(k+1)}$ denotes the distance between consecutive waypoints along that route. The symbol "min" represents the action of finding the smallest value among all the routes evaluated in a specific iteration of the GA. The goal is to identify the path that minimises the overall distance, and "minDist" represents the minimum value attained during the execution of the algorithm. The visualisation techniques encompass the act of graphically representing progress and exhibiting the ultimate optimised route. From the improved GA algorithm, we will obtain our optimised route. To evaluate the distance obtained by the enhanced GA with the distance collected from the TSP library database, a percentage is determined using Equation (4). The difference in trip lengths is then presented in result section.

$$Trip\ Distance\ Difference = \frac{TSP\ library\ outcome - GA\ outcome}{GA\ outcome} \times 100 \quad (4)$$

The Genetic Algorithm requires the following steps, which are explained below.

- Evaluation.
- Crossover.
- Genetic mutation.

The GA employs a population of distinct entities, each with an associated measure of fitness, to advance towards a subsequent generation of the population. This algorithm is inspired by the principles of human evolution proposed by Darwin, as well as the occurrence of naturally occurring genetic processes such as crossover (recombination) and mutation. An individualised approach can be found for each member of the population to address a particular problem. GA aims to solve problems effectively and accurately by genetically reproducing a group of individuals.

Evaluation: The aspect of evolution is crucial in GA. Here, we employ the probabilistic permutation technique to evaluate the quality of a chromosome. The number of beginning chromosomes (parents) is determined based on the dimension of the population, and their overall distance is calculated.

Crossover: After the completion of the selection process, the chromosomes of the parents are merged to create new chromosomes. The next set of children is considered a new path here; the outcomes of searches may be significantly influenced by the combination and by the crossover system.

Mutation: The process of mutation enables the creation of an additional system. The method commences by selecting a parent chromosome according to the shortest distance from the population. An arbitrary location throughout the sequence is chosen and the element at that position is changed. The modified sequence is then copied to the following iteration. We selected the minimal distance value from the chromosomal site at random to mutate and alter its value. There exist numerous categories of mutation drivers. Three forms of mutation processes are employed here. The first type is the flip mutation, which is applicable exclusively to binary chromosomal representations. It involves flipping the chromosome from one place to another. The second process is the swap operation, which allows for the exchange of one place with another location within the chromosome. The final step is the sliding procedure, in which the chromosome is moved from one point to another to determine the route with the least distance for the TSP. This operation proceeds until the given number of iterations has been accomplished.

GA Algorithm 1 pseudocode is given below:

After obtaining the optimised path from the GA, we present a geometrical-based obstacle detection and avoidance (QuickNav) method that is fast-converging, extremely scalable, and operationally cost-effective. Following the completion of the optimal path, the GA generates this path as a series of waypoints. These waypoints are then used as one of the key inputs for the QuickNav algorithm. In addition to the optimised course, QuickNav also relies on the coordinates of known obstacles in order to carry out its role successfully. The incorporation of these two algorithms is essential for the path planning process. After GA finishes its computation by identifying the optimal sequence of waypoints, these data

are inputted into QuickNav. The QuickNav algorithm uses data from both the GA-derived path and obstacle location to continuously alter the trajectory of the UAV. This ensures that the UAV avoids obstacles while still following the most efficient route. This process is performed off-board, allowing for comprehensive analysis and adjustment before the actual flight mission. The smooth transfer of data from the GA to QuickNav, as shown in Figure 2, demonstrates a systematic process of combining optimised route data. This process improves the operational safety and efficiency of the UAV. This approach is ideal for optimising a route at a fixed altitude. Figure 3 represents multiple obstacles together with a varying number of waypoints. In this example, the QuickNav algorithm employs square areas centred around each obstacle location coordinates to reduce the complexity of the route planning procedure. These square zones are assumed to cover the obstacles, despite their actual shapes or sizes. The UAV safely navigates across these square zones, avoiding direct contact with the enclosed obstacles. This method is especially efficient in regulated indoor environments where the size of obstacles is constant and can be predetermined.

Algorithm 1: Enhanced Genetic Algorithm path planning pseudocode

Start or set up:

a. **Waypoints:** *A collection of coordinates (x, y) that indicate specific destinations.*
b. **Max_iterations:** *The upper limit on the number of iterations that the Genetic Algorithm (GA) will perform.*
c. **Population size:** *The quantity of chromosomes (paths) present in each generation.*

Features:

- **Distance:** *The function distance (p1, p2) computes the Euclidean distance between two waypoints.*
- **Fitness(chromosome):** *Computes the overall distance covered by the chromosome.*

Main iteration:

1. **Create the first population:** *Generate a set of chromosomes with a population size, where each chromosome represents a permutation of waypoint indices ranging from 1 to the total number of waypoints.*
2. **Iterative Loop:**
   - **Time:** *For every iteration i ranging from 1 to maximum iterations:*
   - **Fitness Evaluation:**
     - ➢ *For every chromosome c in the population:*
     - ➢ *Determine the overall distance of the path using the fitness function (c).*
   - **Selection:** *Choose a group of chromosomes (parents) for crossover based on their fitness (specifically, the shortest path length).*
   - **Crossover:** *Utilise the crossover operator, such as single-point crossover, to produce offspring chromosomes containing rearranged waypoint sequences.*
   - **Mutation:** *Randomly alter offspring by introducing mutations, such as switching waypoint indices, with a low chance.*
   - **Updated:** *Merge the chromosomes of parents and kids to create the population of the following generation.*
3. **Discover Optimal Route:** *Determine the chromosome in the final population that has the smallest path length.*

The output:
Best chromosome: *The chromosome that represents the most optimal route for the UAV without considering obstacle avoidance.*

The obstacles are identified as "O_n", where n = 1, 2, 3. . . is the number of obstacles in a TSP model.

The suggested approach requires the use of a simulation or numerical technique to identify and overcome obstacles present in the surrounding TSP environment. To simplify the scenario, it is assumed that any obstacles that are visible here are in the form of square shapes and the size of the obstacle always remains inside of the square region. The dimensions of the obstacles are determined by the perimeter necessary for the UAV to

navigate safely. A series of UAV waypoints and obstacle coordinates are obtained from the TSP model via a motion capture system. Figures 4 and 5 show the obstacle detection process and geometric perspective in determining the avoided path between waypoints when an obstacle is identified.

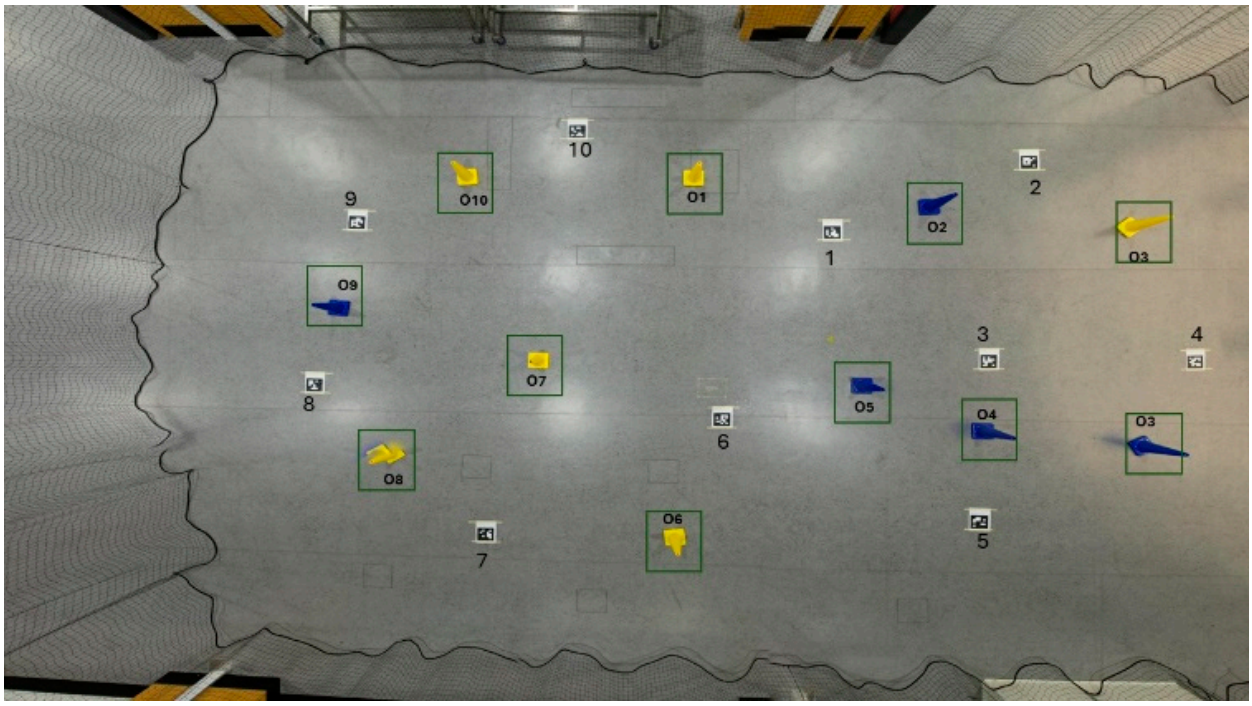

Figure 3. TSP model with obstacles (top view), where the cones represent the obstacles and marker points indicate the visiting waypoints for UAVs.

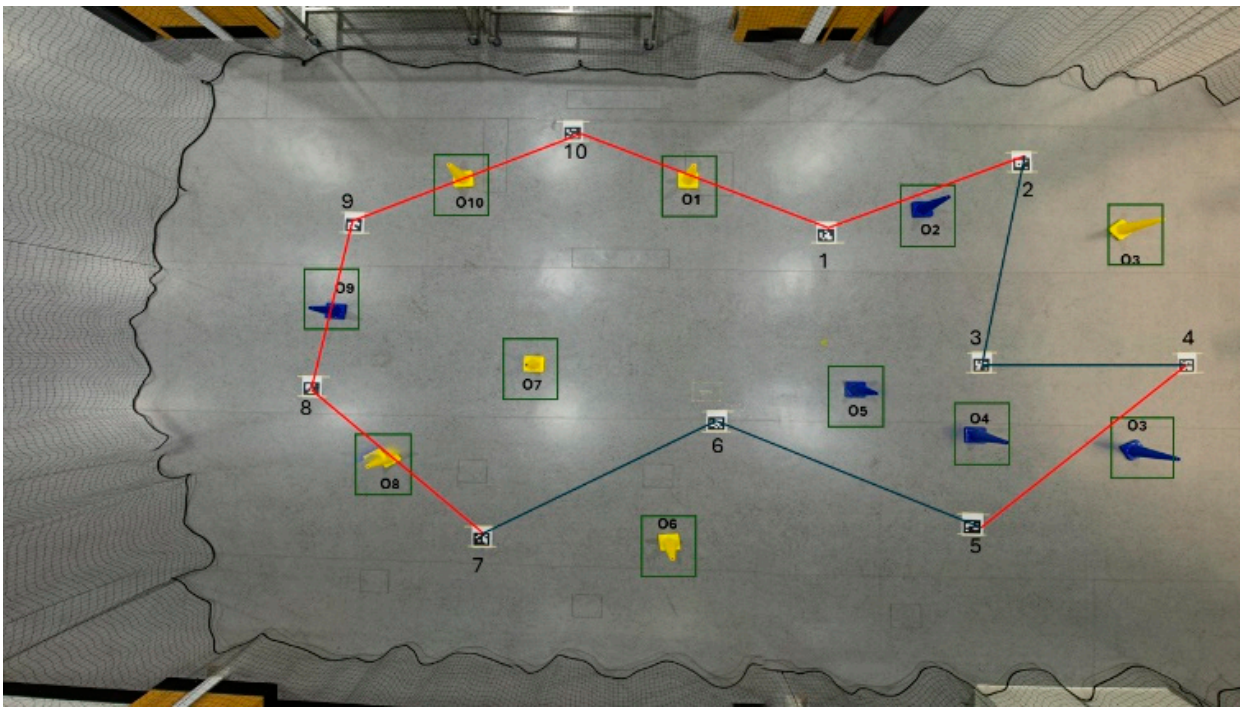

Figure 4. TSP model with obstacles (top view), where the cones represent the obstacles and marker points indicate the visiting waypoints for UAVs (red line indicates obstacles between waypoints and green line indicates obstacle free path between waypoints).

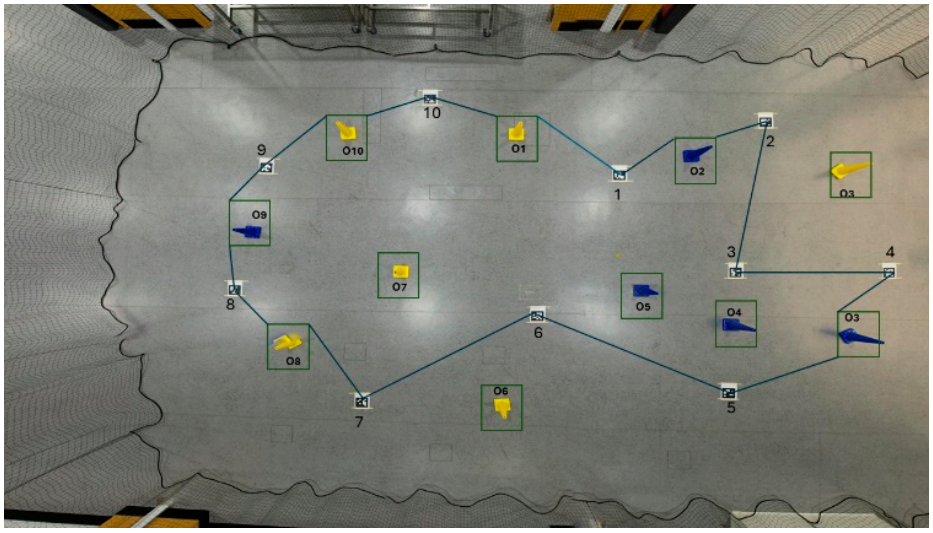

Figure 5. QuickNav model for obstacle avoidance in a TSP model (top view), where the line indicates the way obstacles were avoided along a path where obstacles were detected.

On the path between the two visiting waypoints (VP), the algorithm can encounter obstacle O_1, as shown in Figure 6. The presented detection approach employs linear equations to identify the path obstacle by determining the locations of intersections between the perimeters of the obstruction and the route lines. Any barriers encountered along the visiting points VP_1-VP_2 lines will be seen as obstacles and must be avoided. The inevitable trajectory is depicted as a linear, unbroken line, whereas the potential alternative courses are illustrated by dashed lines. In this situation, the avoidance strategy will involve at least two separate approaches to circumvent the obstruction O_1. These routes will be around the borders of the obstacle, which are designated as avoidable points (ap). Each obstacle will have four avoidable points, denoted as ap_1, ap_2, ap_3, and ap_4. The path VP_1-VP_2 intersects the fences over obstacle O_1, indicated by the red x. Therefore, there are at least two alternative routes that can be taken (shown by the dashed lines) in the form of VP_1–O_1 ap_1)–$O_1(ap_2)$–VP_2 and VP_1–$O_1(ap_3)$–$O_1(ap_4)$–VP_2. The shortest route from VP_1 to VP_2 is used as the requirement for selection. The shortest way in this example is VP_1–$O_1(ap_1)$–$O_1(ap_2)$–VP_2, and as such, it is chosen as the avoidable path.

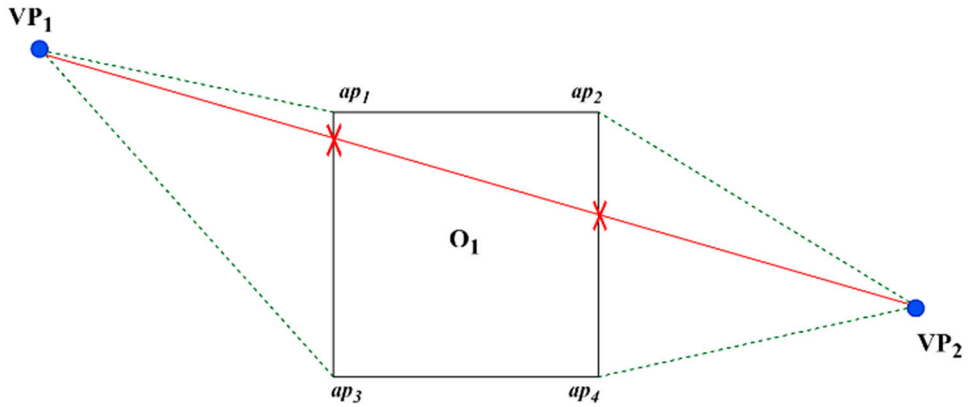

Figure 6. Avoidance route in a TSP's visiting points.

The algorithm evaluates the trajectory and identifies obstacles for each path iteration, then estimates the path with the shortest length. By applying this methodology, we propose a detection method based on the intersection of two diagonal perpendicular lines at the centre point of an obstacle. Following that, we divide the two lines into four separate lines, each extending from the centre of the obstacles to each corner of the square, which we have

specified as the locations that we want to avoid. The APs are extended from the obstacle point by a secure perimeter, which is described as r. Here, r represents the horizontal and vertical distance from the centre coordinate of the obstacle. For example, if the path crosses $O_n(R_1)$, $O_n(ap_1)$ will be the waypoint to avoid. By employing this technique, if a route intersects any of the four diagonal lines, the algorithm can promptly acknowledge the corresponding access points as the waypoints to avoid, as represented in Figure 7.

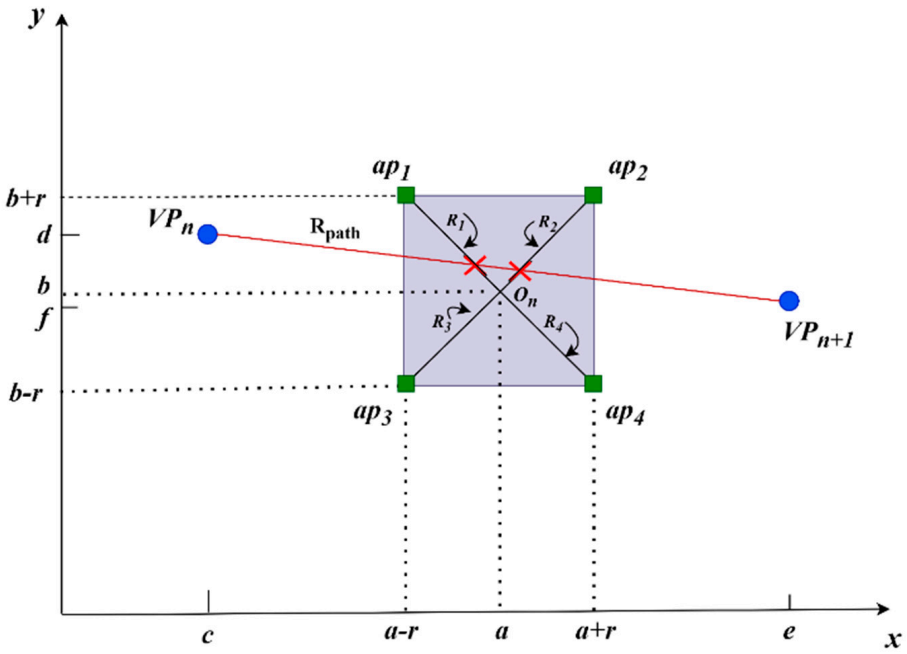

Figure 7. Linear equation-based interception techniques for detection and avoidance.

The algorithm accepts the optimised waypoints from the GA and the coordinates of the obstacle centres as inputs. A square zone is generated around the obstruction point, using the safe distance around the obstacle as the value of r. Four diagonal points are generated for each obstacle, creating a square zone that is shaded, as shown in Figure 7. The information is provided with the coordinates of the obstacle. Around the obstacle's central point, $O_n = (a, b)$, the safe flying perimeter around the obstacle centre point is r. The value of r represents the radius that determines the boundary of the safe flying area around the obstacle. It is determined based on the obstacle's central point. The illustration depicts an obstacle, symbolised by a red irregular shape, positioned at a specific spot in Figure 8. The obstacle is encompassed by a red square box, with the width of the box (w) determined by the largest dimensions of the obstacle. The width of the flying perimeter is increased by adding a buffer zone (b) to ensure safety, resulting in $r = w + b$. The expanded buffer distance surrounding the obstruction results in the creation of a green square, which represents the safe zone. This ensures that the UAV can navigate from point A to point B without colliding with the obstacle. The purpose of this buffer is to accommodate any uncertainties in the dimensions of obstacles and the accuracy of UAV navigation, thus ensuring the safe execution of operations. The square zone approach is more efficient than other geofencing methods, such as circular or dynamic geofences. It simplifies computation by creating a uniform buffer around the obstacle, making implementation easier and computationally less challenging. This approach effectively ensures that the UAV stays at a safe distance from obstacles. As the UAV moves from point A to point B, it must stay inside the safe perimeter r. This perimeter includes the obstacle width (w) and the buffer (b). By following this clear path, the UAV can prevent collisions and stay within safe bounds while effectively avoiding obstacles.

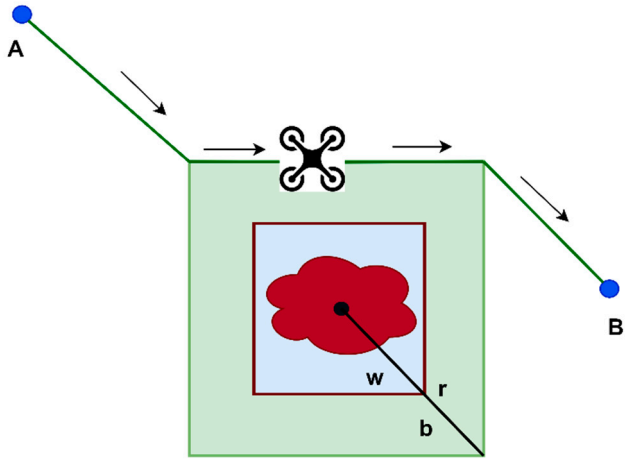

Figure 8. Visualisation of the safe flying zone for UAV obstacle avoidance.

The equation of a line connecting two points (x_1, y_1) and (x_2, y_2) can be expressed using the basic two-point form linear equation as follows:

$$(x - x_1) = \frac{x_2 - x_1}{y_2 - y_1}(y - y_1) \tag{5}$$

The following equation gives the linear distance between all the waypoints.

$$D_{ij} = \sqrt{\sum_{k=1}^{n} \left((x_{ik} - x_{jk}\right)^2 + \left(y_{ik} - y_{jk}\right)^2} \tag{6}$$

In this equation, the variable n denotes the total number of optimised visiting waypoints (VP) from the TSP model, and the summing is conducted across all points ranging from 1 to n.

To establish the flight path between two coordinate visiting waypoints, VP_n (c, d) and $VP_{n+1}(e, f)$, with reference to Figure 6, the following expression can be used:

$$R_{path} = \frac{f - d}{e - c}[(x - c) + d] \; Subject\ to \; \{c \le x \le e\} \tag{7}$$

By implementing the same principles, it can be performed to create four linear equations for each obstacle's coordinate. These equations relate to the obstacle's centre point, denoted as $O_n(a, b)$, and four related avoidable path (AP) values: $[ap_1, ap_2, ap_3, ap_4]$.

$$R_1 = -x + a + b \; ; \; Subject\ to \; \{(a - r) \le x \le a\} \tag{8}$$

$$R_2 = x - a + b \; ; \; Subject\ to \; \{a \le x \le (a + r)\} \tag{9}$$

$$R_3 = x - a + b \; ; \; Subject\ to \; \{(a - r) \le x \le a\} \tag{10}$$

$$R_4 = -x + a + b \; ; \; Subject\ to \; \{(a \le x \le (a + r)\} \tag{11}$$

The decision to avoid obstacles in path planning is required to conform to the following equations.

Lemma 1: A flight trajectory between visiting waypoint VP_n and another visiting waypoint VP_{n+1} is considered to intersect with an obstacle if there exists at least one point of intersection between any line associated with the obstruction.

$$R_{path} = R_1 \; ; Subject\ to \; \{(a - r) \le x \le a\} \tag{12}$$

$$R_{path} = R_2 \; ; \; Subject\ to \; \{a \le x \le (a + r)\} \tag{13}$$

$$R_{path} = R_3 \; ; \; Subject\ to\ \{(a - r) \leq x \leq a\} \tag{14}$$

$$R_{path} = R_4 \; ; \; Subject\ to\ \{a \leq x \leq (a + r)\} \tag{15}$$

Lemma 2: During each iteration, if Lemma 1 holds true, the algorithm must repeat the step to ensure that a new route does not intersect with any obstacle's line after avoidance has occurred. The final avoidable path will be determined by the line that connects the edges of obstacles without intersecting any other lines.

The solution for identifying a new avoidable path (*AP*) when the trajectory intersects with any obstacle lines is given below:

$$AP = \sum_{l=1}^{l=n} VP_l \; R_{path} \sum_{k=1}^{k=n} O_k \; (R_1 \; , \; R_2 \; , R_3 \; , R_4 \;) \tag{16}$$

The value is constrained by the limit specified in Equations (8)–(11) above.

The procedure for solving the given equations is described in Algorithm 2.

In certain circumstances, the act of avoiding one obstacle may result in the avoidance path intersecting with another obstacle. The program will then recalculate in successive iterations until there is no interception for any waypoint in this case. The algorithm compares the current travel line with the lines representing obstacles within the square area defined by two visiting points, such as VP_1 and VP_2, or VP_2 and VP_3, as shown in Figure 9. First, the route linking VP_1 and VP_2 intersects with the $O_1(R_1)$ line. Given that the path intersects $O_1(R_1)$, a new destination point, $O_1(ap_1)$ is designated. A new route has been established connecting VP_1 and $O_1(ap_1)$. Subsequently, the trajectory originating from $O_1(ap_1)$ intersects with the line $O_1(R_2)$, resulting in the emergence of $O_1(ap_2)$ as a novel point that must be avoided before reaching VP_2. Similarly, while transitioning from VP_2 to VP_3, it intersects with the $O_2(R_2)$ line, causing $O_2(ap_2)$ to become a point that can be avoided. Then, as it moves from $O_2(ap_2)$ towards VP_3, though it collided only once during the journey, it will go through another safest path to make sure that it will not collide with the same obstacle again. In this scenario, it will go to $O_2(ap_4)$ before moving to destination point VP_3. Hence, the ultimate avoided route is the route that links VP_1-$O_1(ap_1)$-$O_1(ap_2)$-VP_2-$O_2(ap_2)$-$O_2(ap_4)$-VP_3, as seen by the bold green line in Figure 9. The QuickNav algorithm in Figure 9 ensures that the UAV safely navigates through the safe zone locations, which are defined by the square zones surrounding obstacles. Despite the seemingly shorter distance between O_2-(ap$_2$) and VP_3, the algorithm requires that the UAV must first go from VP_2 to (ap_2), then to (ap_4), and lastly to VP_3. This trajectory ensures that the UAV remains at a secure distance from the obstacle while following the safety limitations set by the specified square areas. QuickNav is specially designed to navigate via a minimum of two secure areas if it detects obstacles along any route, significantly improving the safety of the UAV. The approach focuses an emphasis on avoiding collisions and assuring safety. It ensures that the UAV effectively navigates around objects while maintaining a distance to prevent collisions.

This method continues until all the routes on each waypoint are free of any intersections with the obstacle lines. After the process finishes, the avoided path is graphed for presentation. The following chapter will evaluate the algorithm's effectiveness in terms of reducing the distance and demonstrate it in various scenarios. Furthermore, we calculate the travel time and the durations of the alternative paths. In this study, we define r as equivalent to 0.6 units. Therefore, the distance between the obstacle point and the horizontal and vertical boundaries is precisely 0.6 units for each.

Algorithm 2: Geometrically based obstacle detection and avoidance algorithm (QuickNav) pseudocode

Initialise
input *Optimise Visiting points* VP_n *(c, d), where n = 1,2,3. . .*
input *obstacle coordinate, 0n (a, b), where n = 1,2,3. . .*
input *safe flying perimeter r,*
define *starting visiting point = [*VP_1*]*
define *destination visiting point = [*VP_2*]*
define *starting path = [Rpath] (between starting visiting point and destination visiting point)*
formulate four functions for every obstacle, O_n{*R1, R2, R3, R4* }
define *current path = starting path*
define *current visiting point = starting visiting point*
Main
while *collision detected! Check*
if *<current path intercepts* R_1 *>*
update *{next visiting point candidate} =* ap_1
else if *<current path intercepts* R_2 *>*
update *{next visiting point candidate} =* ap_2
else if *<current path intercepts* R_3 *>*
update *{next visiting point candidate} =* ap_3
else if *<current path intercepts* R_4*>*
update *{next visiting point candidate} =* ap_4
break;
find *the nearest distance between {next visiting point candidate} and*
[current visiting point]
update *[nearest {candidate next visiting point}] = [current visiting point]*
end if
while *no collision was detected! do*
update *[current visiting point] = [destination visiting point] //move to the next path*
update *[destination visiting point] = destination visiting point ++*
update *avoiding waypoint = {starting visiting point,. . . end visiting points}*
end

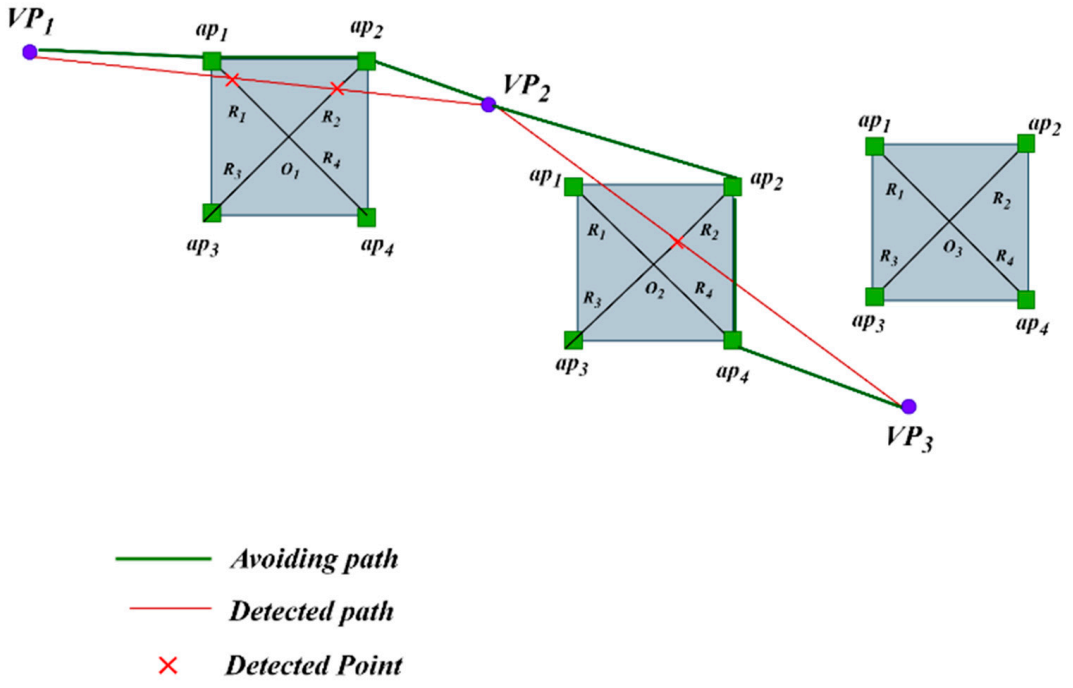

Figure 9. A demonstration of the algorithm's ability to resolve many obstacles.

3. Results

To evaluate the efficiency of our proposed enhanced Genetic Algorithm, we conducted experiments using four scenarios sourced from the TSP database [39]. Each scenario in the dataset presented a distinct level of complexity. We utilised TSP examples from the TSP database package to evaluate the performance of our GA-QuickNav algorithm. The TSP database offers a comprehensive set of sample examples for the TSP and related issues, which have gained significant popularity in the area. In general, a combination of exact and heuristic methods is used to find answers to these problems. The branch-and-cut algorithm is commonly utilised for the solutions. This method utilises the ideas of branch-and-bound, along with cutting planes, to progressively enhance the solution space until the optimal solution is achieved. Furthermore, the Concorde TSP Solver, known for its effectiveness and precision, has been applied to solve numerous TSP database problems that have demonstrated optimality. By utilising well-established, optimal solutions, the comparisons used in our study provide an efficient foundation to assess the performance of our proposed method. Table 1 displays a comprehensive list of difficulties together with their corresponding total location points. The TSP library contains a set of TSP problems together with their optimal solutions, allowing for comparison and evaluation. The items in Table 1 were chosen according to the level of intricacy. The naming convention of the problems specifies the name of a city and the number of waypoints. For instance, BAYG29 is a specific route within the city of BAYG that includes a total of 29 cities.

Table 1. Listed problems from the TSP library.

Name of the Problem	Total Number of Waypoints
BAYG29	29
EIL101	101
GR202	202
PA561	561

The location of points data are uploaded into the GA for optimisation purposes, based on the selected problem from Table 1. The result obtained using GA is depicted in Figure (10-13)b for each problem, throughout the images from Figures 10–13. Figure 10 shows identical results for BAYG29 with a distance of 9073 m using both the TSP library and the GA. The second one is named the EIL101 area scenario, and the provided result is 629 m. As seen on the right side of Figure 11b, the GA calculates a total distance of 639 m. In the next scenario, the scenario is identified as GR202, and the result is 547 m. The GA obtains an outcome of 490 m for the scenario depicted in Figure 12. The outcome of the GA demonstrates that the algorithms produce better outcomes compared to the solution provided by the TSP library. In the last test, there is a selection of 561 cities available, and a specific answer has been provided for this, reaching 19,311 m. As illustrated in Figure 13, the GA computes the final distance of 15,928 m in a more efficient manner compared to the provided distance. In this situation, the result is more effective than the solution provided by the TSP library because of the utilisation of the GA. Figure 13a displays many intersections at the corresponding points of the TSP library approach whereas the GA offers an enhanced and optimal solution by ensuring that there is no overlap or crossover within location points in the process.

As shown in Table 2, all the obtained outcomes are measured against the results collected from the TSP database. The trip distance difference illustrates the proximity of the GA to the outcome of the TSP library data. Positive numbers indicate that the GA solution performs better than the results produced by the TSP Library, showing its greater effectiveness. Conversely, negative numbers indicate situations where the GA solution performs less optimally than the solution provided by the TSP Library. The solutions to the problem BAYG29, both GA and TSP results, are identical, and their percentage levels

correspond to those displayed in Table 2. The GA solution required a total of 3.03 s. Table 2 also demonstrates that the EIL101 GA problem obtains outcomes that are nearly identical to those achieved using the TSP library, with a trip distance difference percentage approaching 0 per cent in just 38.85 s. Other problems, including GR202 and PA561, have GA outcomes that are up to 10.42% and 17.51% higher than TSP database answers, respectively, and both GA results need less than two minutes to resolve the problem.

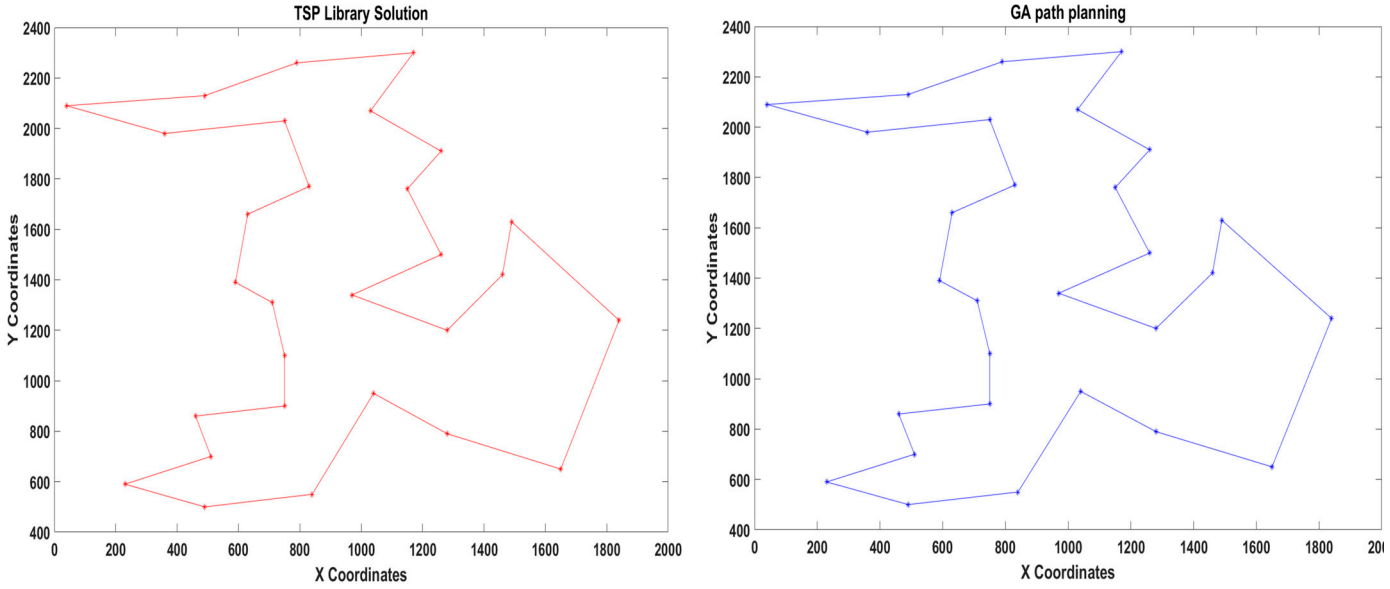

Figure 10. BAYG29: (**a**) Solution using the TSP library. (**b**) Optimised result using a Genetic Algorithm.

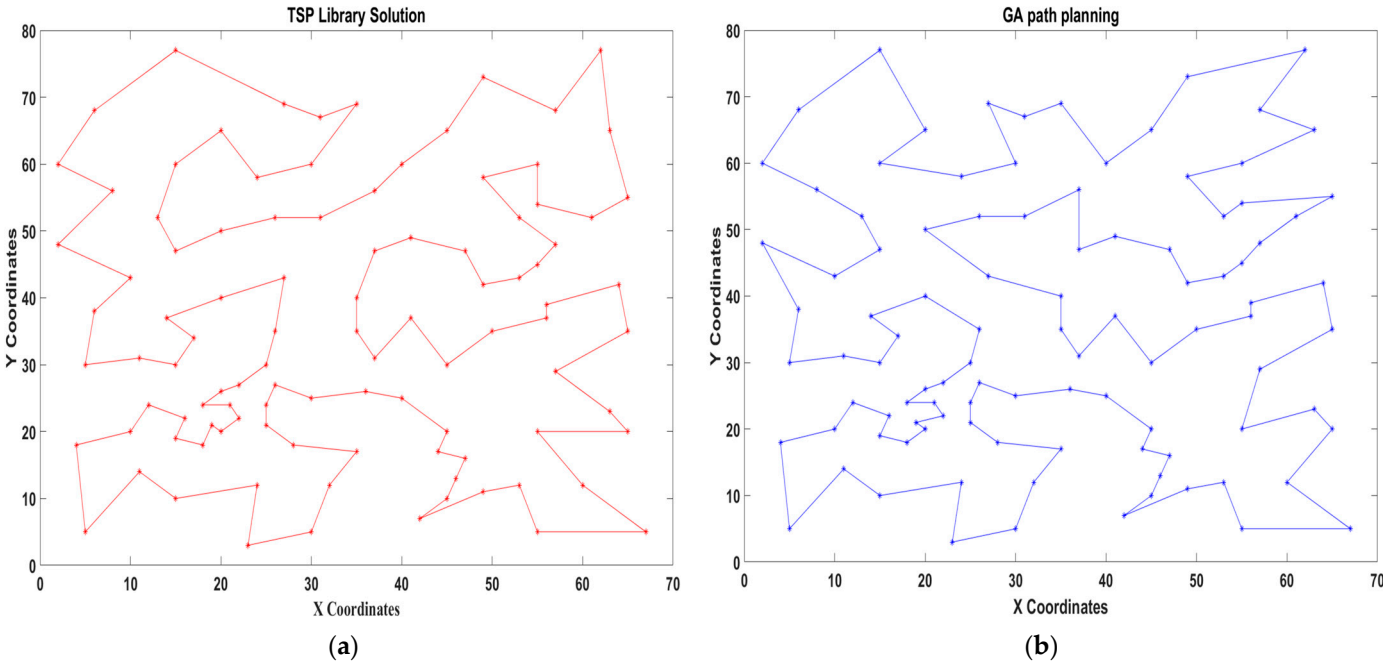

Figure 11. EIL101: (**a**) Solution using the TSP library. (**b**) Optimised result using a Genetic Algorithm.

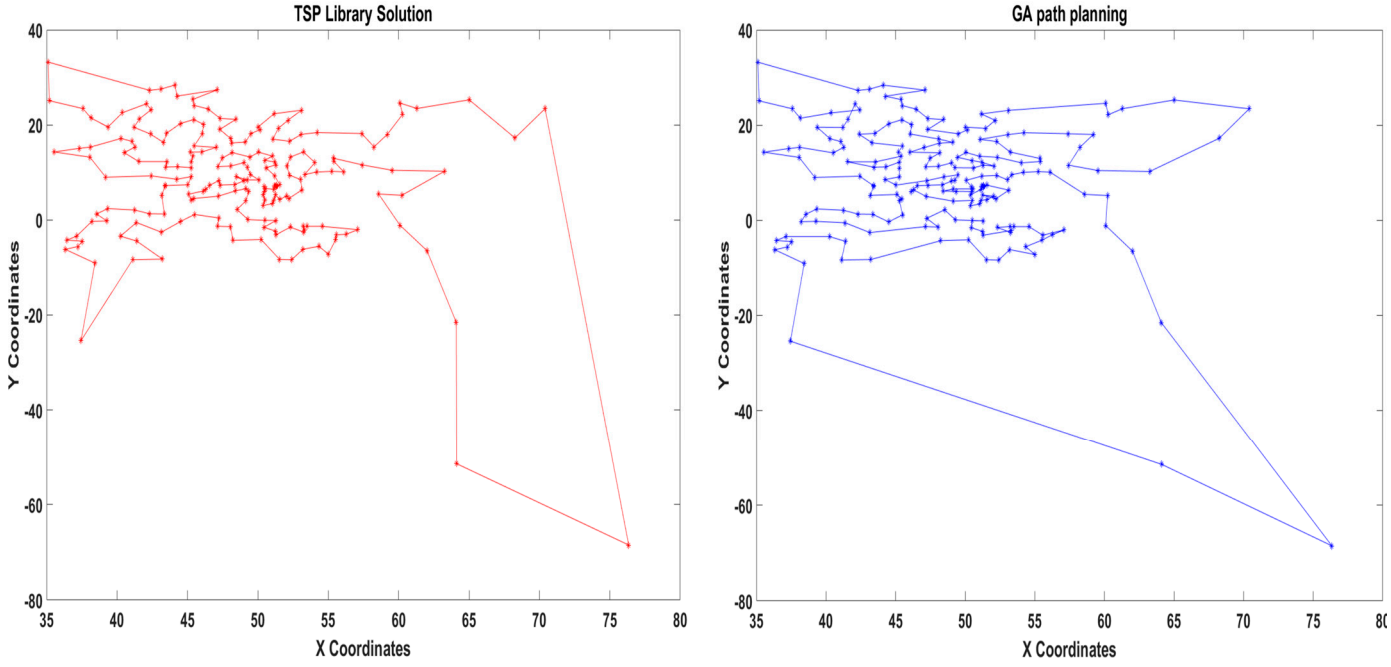

Figure 12. GR202: (**a**) Solution using the TSP library. (**b**) Optimised result using a Genetic Algorithm.

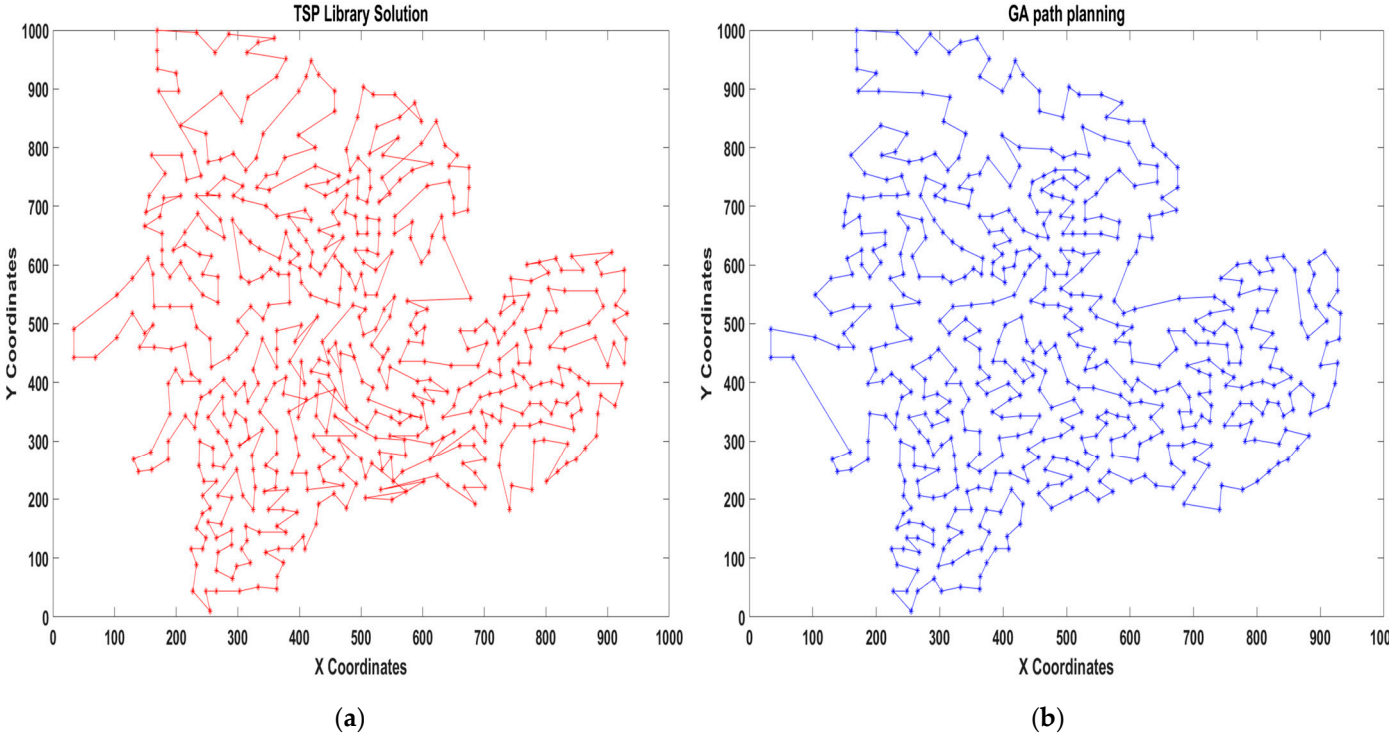

Figure 13. PA561: (**a**) Solution using the TSP library. (**b**) Optimised result using a Genetic Algorithm.

In order to assess the precision and consistency of the GA results, we analyse not only Table 2 but also five GA results for every scenario, as represented in Figure 14 (results no. 1 to 5). Based on the GA outcomes, it can be concluded that BAYG29 shows a high level of consistency. Specifically, BAYG29 exhibits perfect accuracy with a zero percentage of trip distance difference between the solutions. The percentage values for EIL101 in the TSP collection were highly comparable to other values, and the percentage levels for

EIL101 were also near the reference value. On each occasion this problem was executed, it consistently exhibited a trip distance difference rate of approximately 4%. Remarkably, the GA demonstrates outstanding efficiency in solving the GR202 and PA561 problems, exceeding the results obtained using the TSP lib by around 10%.

Table 2. Comparative analysis of the outcomes obtained using the TSP library and the GA (positive sign indicates that the GA trip length is better than the TSP library).

TSP Problem Name	TSP Library Result Trip Distance (m)	GA Result Shortest Trip Distance (m)	GA Outcome Time (s)	Trip Distance Difference (%)
BAYG29	9073	9073	3.03	0
EIL101	629	639	38.85	−1.59
GR202	54 7	490	41.66	10.42
PA561	19,311	15,928	90.42	17.51

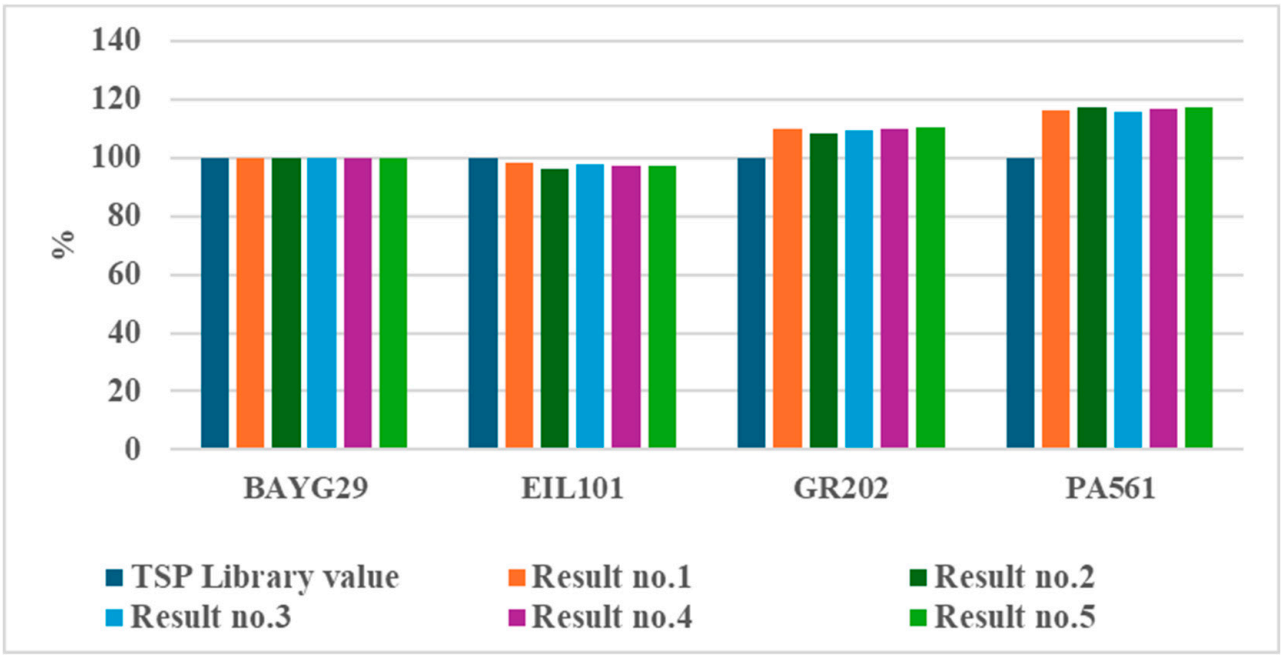

Figure 14. Comparison between the TSP library outcomes and GA results.

Table 3 provides an extensive overview of the distances travelled for specific TSP problems. In the case of the BAYG29 problem, all five outcomes consistently produced a value of 9073. This resulted in a mean of 9073 and a standard deviation of 0, suggesting that there was no variability in the results. Additionally, the 95% confidence intervals (CIs) for both the lower and upper bounds were also 9073, highlighting the algorithm's stability in this situation. Conversely, the EIL101 problem showed little variation, with outcomes ranging from 639 to 653. The average value was 645.8, while the standard deviation was 5.45. The 95% confidence interval for EIL101 ranges from 639 to 652.6, indicating minor variations over iterations. The GR202 problem demonstrated consistent results despite the modest variability, with values ranging from 490 to 502. The mean is 494.8, while the standard deviation is 4.764. The PA561 problem displayed variations, with outcomes varying from 15,928 to 16,300. The mean value was 16101, and the standard deviation was 152.7. These results illustrate how the algorithm responds to different levels of scenario complexity and emphasise the difficulty presented by this more detailed TSP scenario.

Table 3. Statistical analysis for TSP library problem using GA trip distance outcome.

TSP Problem Name	Result no. 1	Result no. 2	Result no. 3	Result no. 4	Result no. 5	Mean	Standard Deviation	95% CI Lower	95% CI Upper
BAYG29	9073	9073	9073	9073	9073	9073	0	9073	9073
EIL101	639	653	642	647	648	645.8	5.45	639	652.6
GR202	492	502	497	493	490	494.8	4.764	488.9	500.7
PA561	16,192	15,977	16,300	16,108	15,928	16,101	152.7	15,911	16,291

Table 4 presents a comprehensive examination of the time required to conduct calculations for the same set of TSP problems. It demonstrates the performance of the GA under various computational requirements. For the BAYG29 problem, computation times were extremely consistent, with a mean of 3.038 s and a very low standard deviation of 0.01304. This demonstrates how well the method works for this scenario. A narrow 95% confidence interval (CI) of 3.022 to 3.054 s indicates predictable and reliable computation times. EIL101 computation times are also efficient, with a mean of 38.98 s and a standard deviation of 0.1153, showing minor fluctuation. This problem's 95% CI is 38.84 to 39.12 s, demonstrating the algorithm's robustness in slightly challenging environments. Computation times varied substantially for the more complex GR202 problem, with a mean of 42.39 s and a standard deviation of 0.631. The 95% CI for this problem, 41.61 to 43.18 s, demonstrates a greater computational time range due to its complexity and processing requirements. The PA561 problem, which is significantly more challenging, has a mean computation time of 91.02 s and a standard deviation of 0.4909. The 95% CI ranges from 90.41 to 91.63 s, suggesting that the method performs well even in particularly complicated circumstances, despite increased computation times.

Table 4. Statistical analysis for TSP library problem using GA computation time outcome.

TSP Problem Name	Result no. 1 (s)	Result no. 2 (s)	Result no. 3 (s)	Result no. 4 (s)	Result no. 5 (s)	Mean (s)	Standard Deviation	95% CI Lower	95% CI Upper
BAYG29	3.03	3.03	3.04	3.06	3.03	3.038	3.022	3.054	0.01304
EIL101	38.85	39.11	38.87	39.01	39.06	38.98	38.84	39.12	0.1153
GR202	41.89	43.06	42.99	42.36	41.66	42.39	41.61	43.18	0.631
PA561	91.46	90.66	91.55	91.02	90.42	91.02	90.41	91.63	0.4909

The performance of the GA-QuickNav algorithm, which combines a Genetic Algorithm (GA) with the QuickNav algorithm, was evaluated using four different TSP scenarios of increasing levels of complexity. We utilised MATLAB for simulations and the Robot Operating System (ROS) platform for conducting real-time flight testing with a UAV. The system architecture is shown in Figure 15.

Flight testing was conducted in real time using the following setup:

- Flight Platform: A Holybro X500 V2 multi-rotor system with 2216 KV920 motors and 1045 propellers (10″/254 mm length, 4.5″/114.3 mm pitch) utilised as the UAV frame shown in Figure 16. The UAV is controlled by a Holybro Pixhawk 6C autopilot running the PX4 firmware. Holybro 915 MHz telemetry radios enabled communication. The 500 × 215 × 1440 mm UAV has a 1-kilogram payload capacity. A four-cell 5000 mAh LiPo battery gave it 8 min of flight autonomy with a payload [40].
- Onboard Computing: The onboard computer for path planning execution used an NVIDIA Jetson Orin Nano developer kit, which consisted of an NVIDIA Ampere architecture featuring 1024 CUDA cores and 32 Tensor cores. It also included a 6-core Arm Cortex-A78AE v8.2 64-bit CPU and 8GB of LPDDR5 DRAM.
- Software Environment: The companion computer is equipped with Ubuntu 20.04, a 64-bit Linux-based operating system (OS). The ROS Noetic version enabled com-

munication between the VRPN-optitrack system, MAVROS, and a waypoint motion command ROS node. To control the UAV, the open-source PX4 firmware version 1.13.2 is used.

- Testing Location: The real-world flight experiments were carried out within an indoor environment at QUT O-134, which has a motion capture system (optitrack) facility for safety and millimetre precision localisation.
- Flight Execution: The GA-QuickNav produces an output flight path as a sequence of waypoints and time for each waypoint. This flight path is then executed by an ROS node communicating with the PX4 firmware through MAVROS an ROS wrapper of the Micro Air Vehicle Link (MAVLink) protocol.
- Indoor Localisation: Optitrack system for localisation connected to the UAV companion computer via WiFi. The companion computer runs a Virtual-Reality Peripheral Network (VRPN) server to obtain the UAV pose in the flight area local frame. This is then transmitted to the Flight Controller Unit (FCU) using ROS and mavros for proper localisation in a GPS-denied environment.

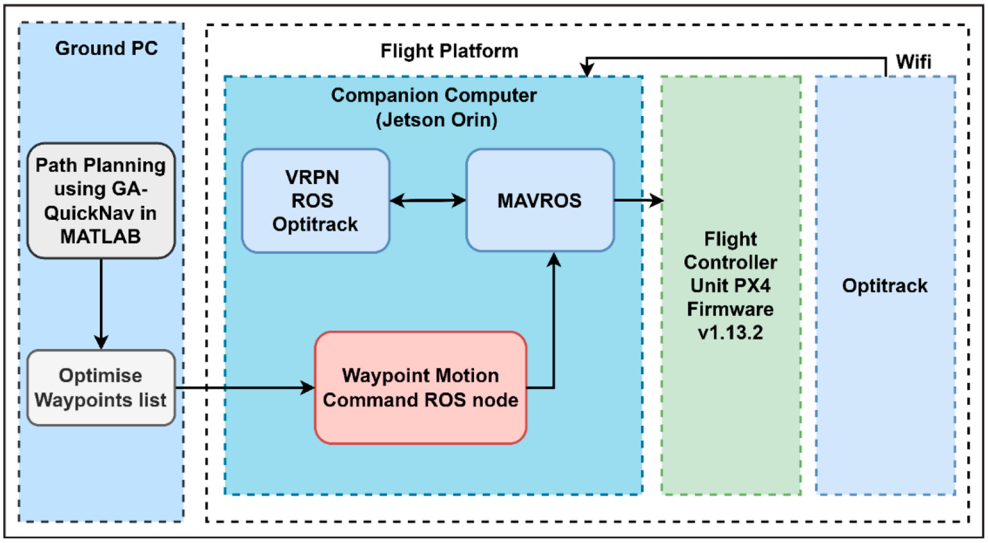

Figure 15. System architecture for autonomous UAV navigation in a GPS-denied environment.

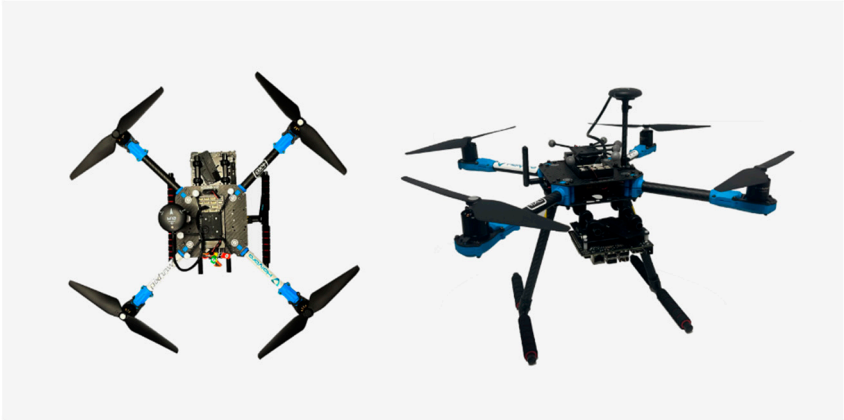

Figure 16. UAV frame used in this research.

Performance criteria:

- Path Optimisation: The algorithm's efficacy in optimising the path is assessed by comparing the total distance covered before and after obstacle avoidance through real-time motion capture system data.

- Computational Time: The GA-QuickNav algorithm's total computational time for processing each scenario was measured.

The outcomes of the simulation flight tests are shown in Table 3.

Table 5 presents the outcomes of a comprehensive study focused on the TSP across several scenarios involving visiting waypoints (VPs) and obstacles. The algorithm was able to detect three of the obstacles in the optimised path during the first experiment, which consisted of ten visiting waypoints and five obstacles. This resulted in a significant reduction from the initial distance of 45.48 m, which was before path optimisation, to the final distance of 28.01 m of the optimised path in a short amount of time, 2.99 s, which is shown in Figure 17. During the second case, which included twelve VPs and six obstacles, the effectiveness of the approach was demonstrated by the fact that the starting distance of 52.66 m was reduced to 29.84 m after optimisation, and the estimated amount of time required for the computation was 2.36 s, as shown in Figure 18. During the third experiment, which consisted of seven VPs and seven obstacles, it was discovered that the optimum route contained five obstacles. The starting distance was 32.09 m, and the final distance was 25.60 m. It took 2.35 s to perform the computations. In Figure 19, the flight path is displayed. According to the results of the last experiment, which consisted of four VPs and eight obstacles, the optimal path was found to have six obstacles. The initial distance was 17.75 m, and after six obstacles were found in the optimum route, the final distance was 21.94 m. The entire processing time was 2.46 s. The flight path is depicted in Figure 20. A video demonstration of the UAV flight test can be accessed at https://bit.ly/UAV-test-video (accessed on 10 May 2024).

Table 5. TSP simulation flight test outcomes using GA-QuickNav.

TSP Problem	Number of Waypoints	Number of Obstacles	Detected Obstacles	Initial Distance (m)	Final Distance (m)	Time (s)
1	10	5	3	45.48	28.01	2.99
2	12	6	1	52.66	29.84	2.36
3	7	7	5	32.09	25.60	2.35
4	4	8	6	17.75	21.94	2.46

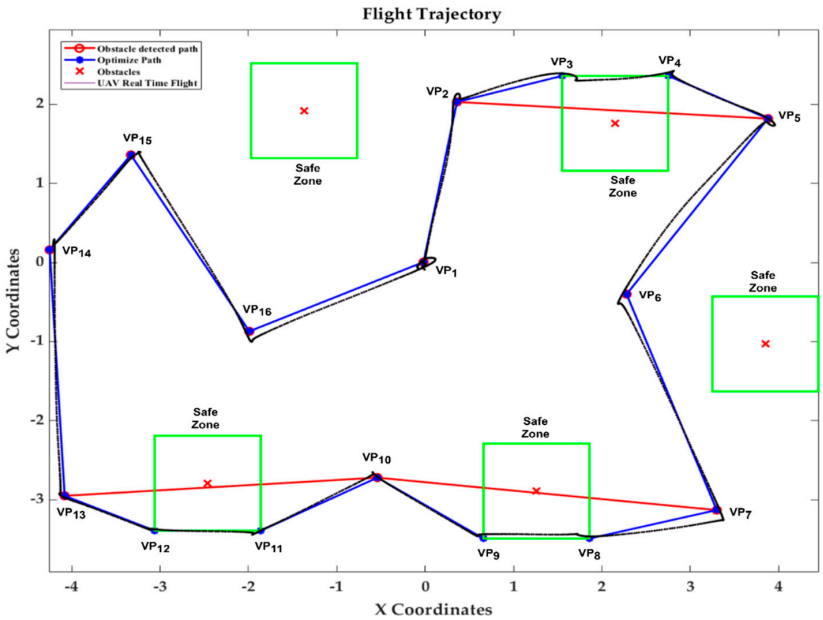

Figure 17. Trajectory representation of Scenario 1, where the red path is determined using GA, the blue path is the trajectory after QuickNav's obstacle avoidance, and the black line represents the real flight path.

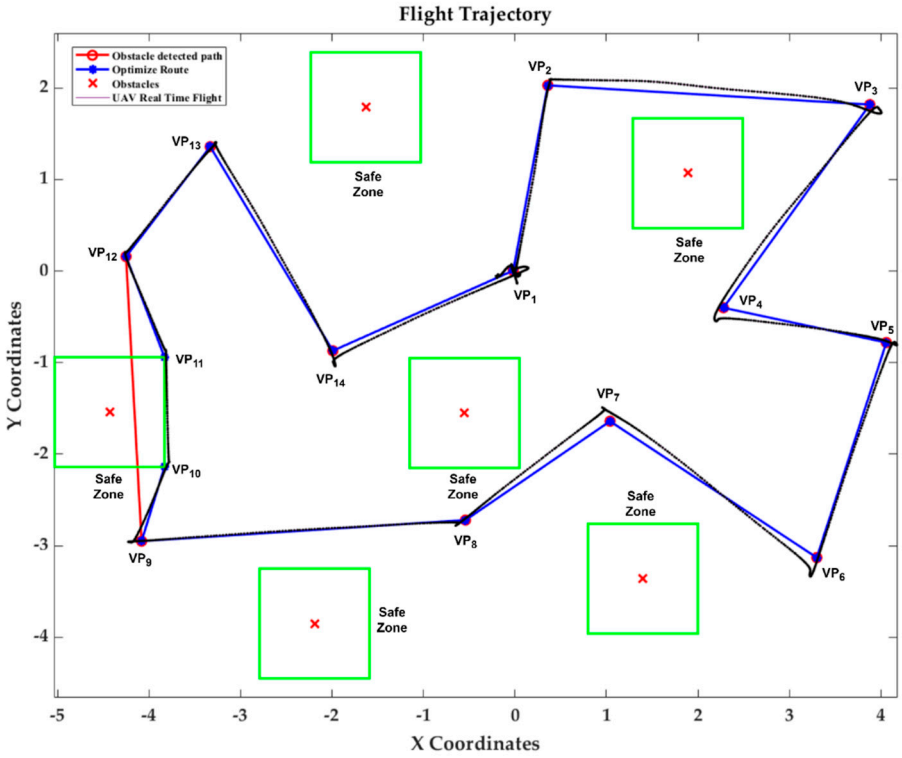

Figure 18. Trajectory representation of Scenario 2, where the red path is determined using GA, the blue path is the trajectory after QuickNav's obstacle avoidance, and the black line represents the real flight path.

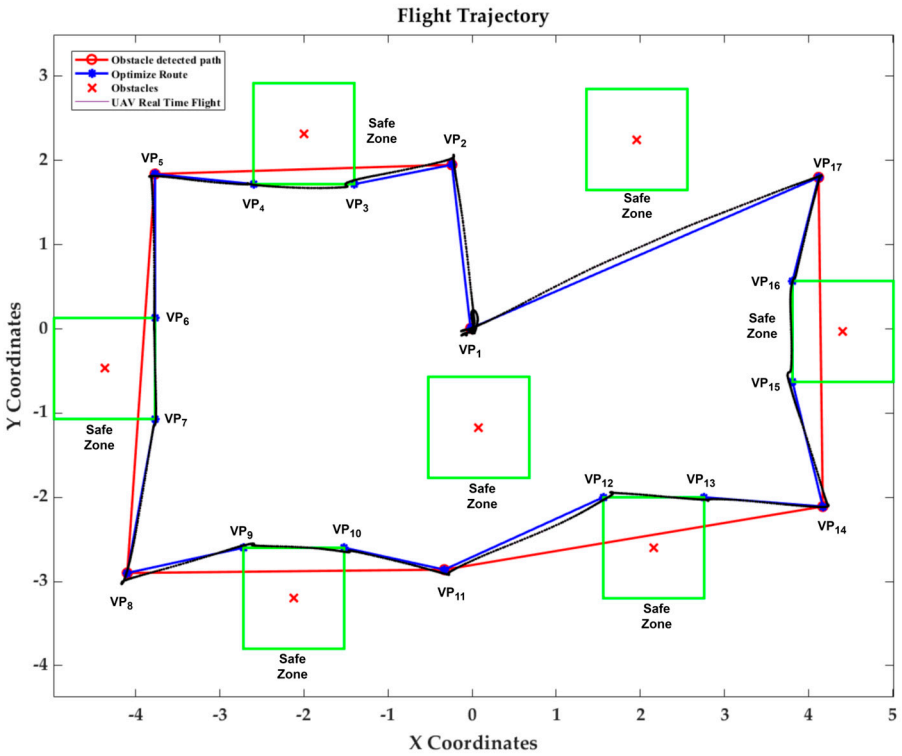

Figure 19. Trajectory representation of Scenario 3, where the red path is determined using GA, the blue path is the trajectory after QuickNav's obstacle avoidance, and the black line represents the real flight path.

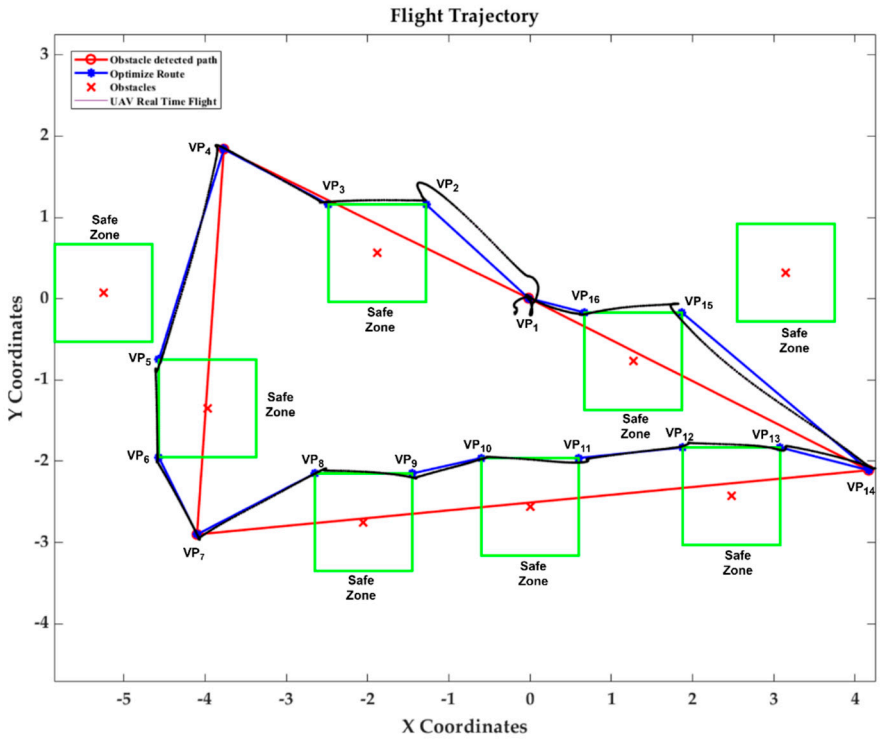

Figure 20. Trajectory representation of Scenario 4, where the red path is determined using GA, the blue path is the trajectory after QuickNav's obstacle avoidance, and the Black line represents the real flight path.

The following Table 6 presents a comparison of the final distances achieved by the three different path planning algorithms, GA-QuickNav, A*, and Nearest Neighbour (NN), across four different TSP problems with different numbers of waypoints.

Table 6. Comparison between TSP simulation flight test trip distance outcomes.

TSP Problem	Number of Waypoints	GA-QuickNav Final Distance (m)	A* Final Distance (m)	NN Final Distance (m)
1	10	28.01	32.48	40.95
2	12	29.84	39.79	38.95
3	7	25.60	26.99	43.58
4	4	21.94	21.93	30.54

The GA-QuickNav algorithm outperformed both the A* algorithm and the NN algorithm in solving TSP problem 1, which included 10 waypoints. It reached a final distance of 28.01 m, while the A* algorithm resulted in a final distance of 32.48 m and the NN algorithm had a maximum final distance of 40.95 m. The findings are visually represented in Figure 21, which demonstrates the efficiency of the GA-QuickNav algorithm in this scenario. For TSP problem 2, consisting of 12 waypoints, the GA-QuickNav algorithm exceeded the A* algorithm and the NN algorithm. It achieved a final distance of 29.84 m, while the A* algorithm and the NN algorithm achieved distances of 39.79 m and 38.95 m, respectively shown in Figure 22. The GA-QuickNav algorithm calculated 25.60 m for TSP problem 3. The obtained result was close to that of the A* algorithm, which achieved 26.99 m, and notably superior to that of the NN method, which achieved 43.58 m. The results are displayed in Figure 23. Regarding TSP problem 4, which involved four waypoints, both the GA-QuickNav and A* algorithms produced comparable final distances of 21.94 m and 21.93 m, respectively. Nevertheless, the NN method showed lower efficiency, resulting in a final distance of 30.54 m. Figure 24 illustrates the outcomes, showing that

GA-QuickNav and A* performed similarly in this situation. However, the NN algorithm was unable to navigate around multiple obstacles in a single path, resulting in an inefficient route. In summary, the table and figures demonstrate that the GA-QuickNav algorithm consistently achieves shorter travel distances compared to the A* and NN algorithms in various scenarios with different numbers of waypoints. These results indicate that the GA-QuickNav algorithm is effective and reliable in path planning for UAVs.

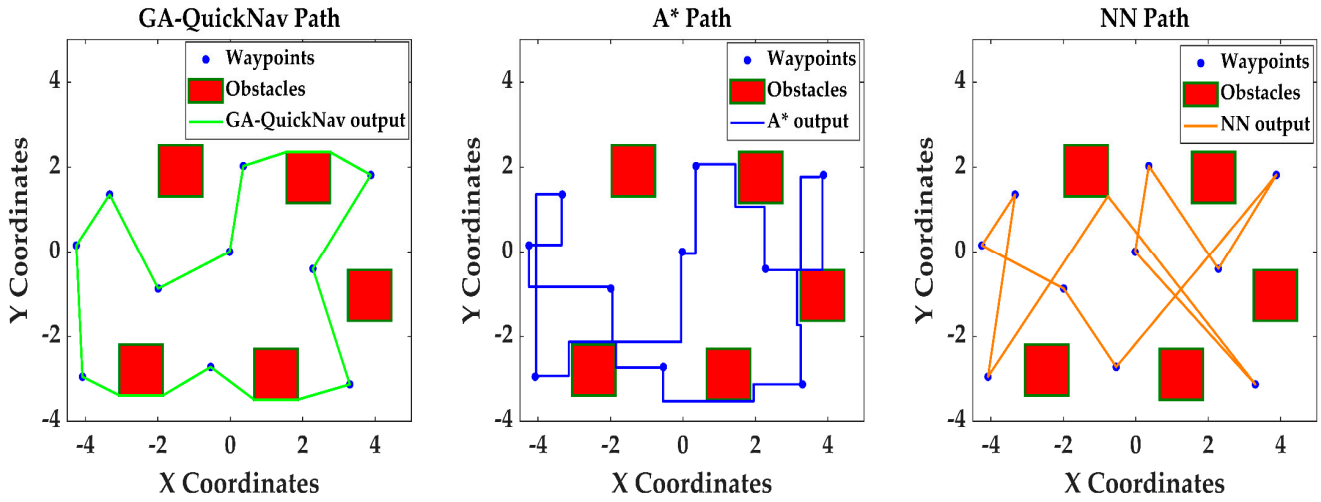

Figure 21. Trajectory representation of TSP problem 1, where the green path is determined using GA-QuickNav, the blue path is the outcome using A*, and the orange line represents the NN flight path outcome.

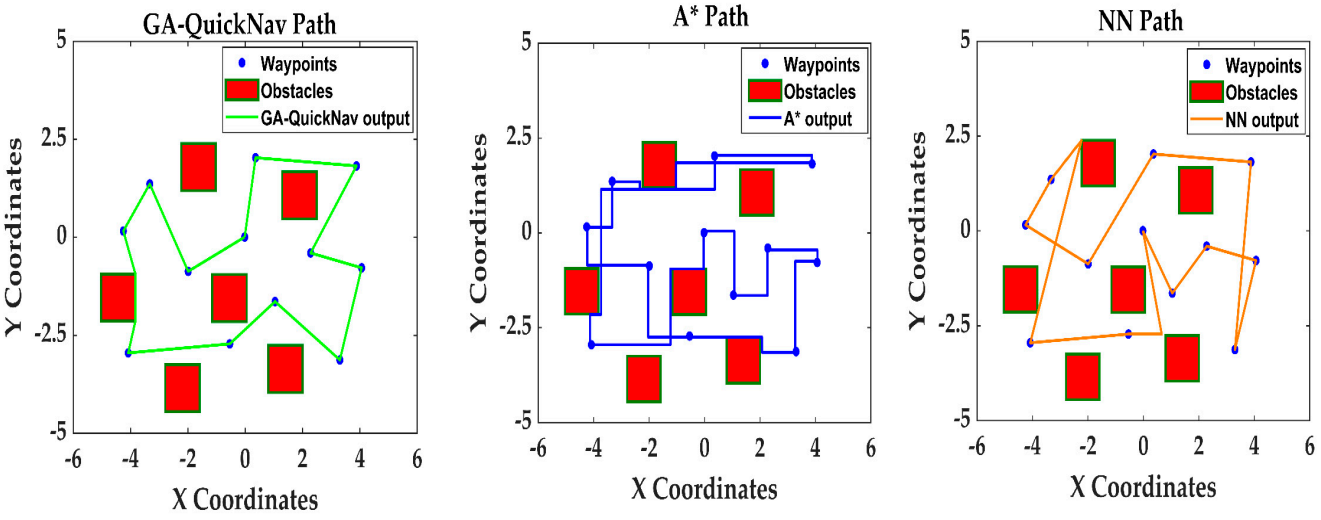

Figure 22. Trajectory representation of TSP problem 2, where the green path is determined using GA-QuickNav, the blue path is the outcome using A*, and the orange line represents the NN flight path outcome.

The proposed combination of the GA-QuickNav approach proved the ability to efficiently optimise paths across a variety of scenarios requiring visiting waypoints and obstacles, according to the thorough results provided in the study. Nevertheless, numerous obstacles and significant discoveries arose from the thorough validation process. The algorithm demonstrated strong performance in optimising pathways while assuring the avoidance of obstacles, as seen by the substantial reductions in total travel distances achieved after integrating the QuickNav module in all test situations (Table 5). In Scenario 1, the starting distance of 45.48 m was decreased to 28.01 m after optimisation and ob-

stacle avoidance, when there were 10 waypoints and 5 obstacles. The algorithm showed its effectiveness in path optimisation when crossing dense areas with obstacles, as seen by the significant reductions that were also noted in the other instances. In addition to four waypoints and eight obstacles, Scenario 4 was a significant challenge, as the ultimate distance of 21.94 m was larger than the initial distance of 17.75 m. This outcome includes a thorough explanation, specifically regarding the connections between the GA and the QuickNav obstacle avoidance system. At first, the GA identified the optimal route by only evaluating the shortest distance between waypoints, without considering any possible obstacles. After this optimisation, QuickNav executed its obstacle identification and avoidance procedure. During this stage, QuickNav identified three obstacles located along the GA's first optimised path shown in Figure 20. To ensure safe navigation, the UAV needed to effectively manoeuvre around these obstructions by following designated safe zones. Particularly, the UAV navigated the shortest path found involving waypoints VP_7 through VP_{14}, effectively avoiding the obstacles while following the safe flying perimeter established for each of them. As a result of this diversion, the UAV ended up travelling through additional areas that were not initially part of the straight route, thereby increasing the total amount of distance travelled. The situation where there are many obstacles compared to the number of waypoints demonstrates the inherent compromise between minimising journey distance and assuring the safe avoidance of obstacles. The algorithm prioritised safety by intentionally diverting from the shortest path in order to effectively navigate around various obstacles. Regardless of this limitation, the suggested method consistently showed its ability to create optimal paths involving visiting waypoints and obstacles across all scenarios assessed, as shown by the representations of the paths (Figures 17–20). Although the algorithm has shown strong performance in tested circumstances, its ability to handle bigger and more complex real-world scenarios has to be further examined. The main focus of the present study was on indoor scenarios with predetermined obstacle layouts, which may not fully represent the complexity found in large open environments. Expanding the algorithm to include additional circumstances presents difficulties, including enhanced computational requirements caused by larger amounts of data and more intricate path computations. In order to address these problems, future implementations may employ distributed computing approaches to improve processing capabilities. Furthermore, in order to efficiently handle dynamic obstacles in extensive and complex situations, the algorithm needs to be capable of adjusting to a three-dimensional operational environment. This includes travelling across various heights and addressing aerial obstacles that do not exist in two-dimensional scenarios. These improvements are crucial for maintaining the effectiveness and efficiency of the GA-QuickNav algorithm as the complexity of the task environment increases. These developments are essential for expanding the usefulness of our technique in situations like urban navigation and disaster response operations, where it is necessary to avoid dynamic obstacles and quickly optimise the course. The UAV successfully executed the optimised paths while avoiding obstacles based on real-time localisation data from the motion capture system, further validating the algorithm's practical applicability in real-world flight tests conducted in a GPS-denied indoor environment. Additionally, the incorporation of QuickNav significantly improved computational efficiency, with execution times varying from 2.35 to 2.99 s throughout the scenarios. This computational efficiency is essential for real-world applications, especially time-sensitive situations, which require swift path optimisation with minimal latency while maintaining obstacle avoidance and safety. Overall, the extensive testing procedure emphasised the algorithm's strength, effectiveness, and practicality for planning routes for scenarios that involve travelling to waypoints and avoiding obstacles. This presents the combination of GA-QuickNav as an appealing approach for improving the autonomy and safety of UAVs in real-world operations.

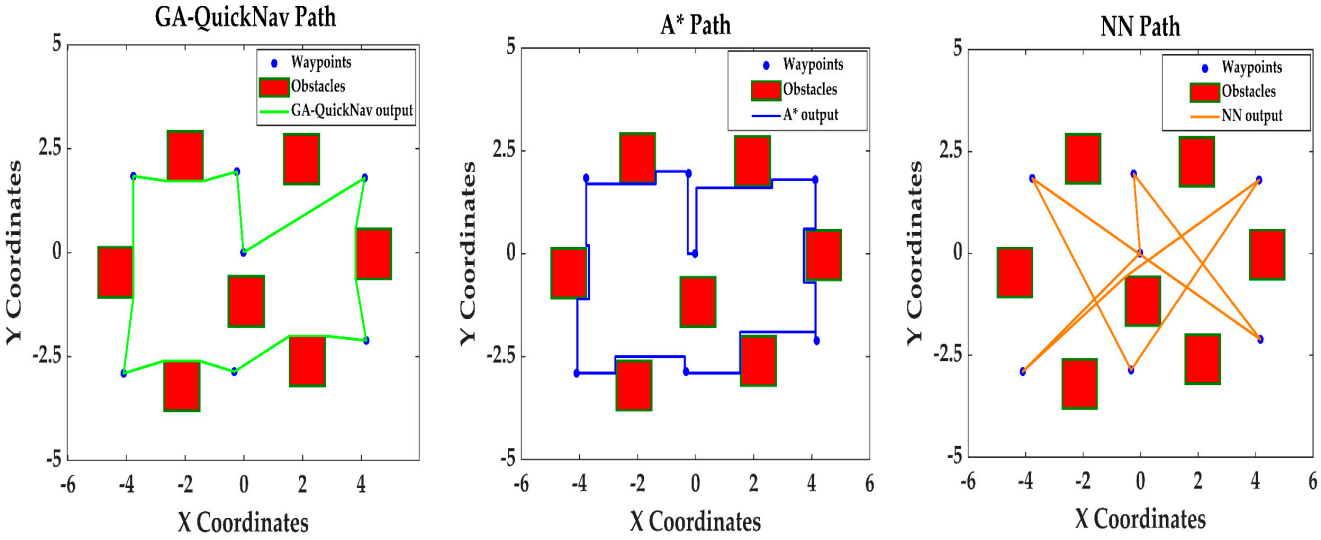

Figure 23. Trajectory representation of TSP problem 3, where the green path is determined using GA-QuickNav, the blue path is the outcome using A*, and the orange line represents the NN flight path outcome.

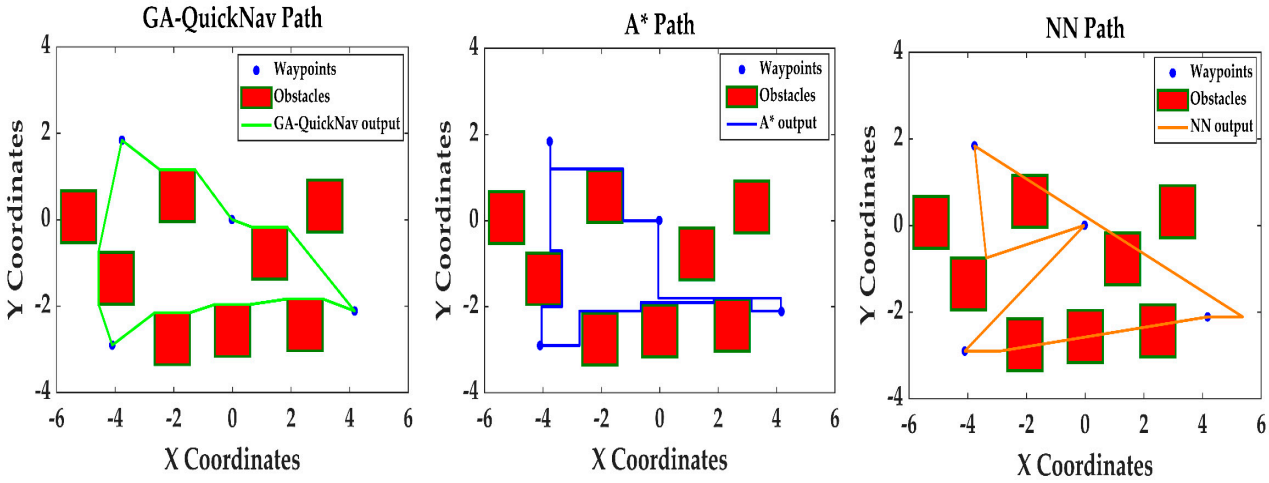

Figure 24. Trajectory representation of TSP problem 4, where the green path is determined using GA-QuickNav, the blue path is the outcome using A*, and the orange line represents the NN flight path outcome.

4. Conclusions

This research presents an innovative combined approach for offline path planning and obstacle avoidance for UAV operations, addressing the key challenges encountered in applications such as Wireless Sensor Networks (WSNs) and precision agriculture. The proposed approach combines an advanced GA with a geometrically based obstacle avoidance algorithm, QuickNav, to optimise mission trajectories and ensure safe and efficient UAV missions. The GA efficiently improves the optimisation of trajectories by generating a population of potential routes and using iterative selection processes to find the most optimal trajectory for visiting multiple predetermined waypoints. Concurrently, the QuickNav algorithm employs linear equations and geometric methods to identify and avoid obstacles along the optimised path, providing collision-free navigation. The integrated GA-QuickNav approach has been extensively evaluated through comprehensive simulations on assessed TSP scenarios and real-world UAV flight tests conducted in an indoor GPS-denied environment using a motion capture system for localisation. The proposed algorithm consistently proved its efficacy in reducing overall travel distances, successfully

navigating UAVs through complex environments, and adhering to safety constraints across diverse scenarios with different waypoint combinations and obstacle densities. Using real-world testing to systematically evaluate and validate the proposed method gives this research credibility and practical relevance. The comprehensive testing conducted in various environments provides significant insights into the algorithm's capabilities, strengths, and potential areas for further improvement. This study assumes that the UAV maintains a constant altitude, which simplifies the problem to a two-dimensional path planning scenario. This assumption is in line with the special requirements of interior environments or fields such as precision agriculture and WSN monitoring. In these cases, having a consistent altitude helps ensure the accuracy and dependability of data gathering. To guarantee consistent coverage, it is necessary for the UAV to maintain a steady elevation, which allows for optimised sensor operations. Nevertheless, we recognise that this assumption limits the relevance of the findings to dynamic outdoor scenarios with large height variations caused by diverse topography and aerial obstructions. However our results are limited in their applicability, this also highlights the opportunity to expand this research into further complex scenarios. Also, the present approach of the QuickNav algorithm involves enclosing obstacles within uniform square zones. This facilitates navigation, but it is specifically designed for environments that have predictable and consistent obstacle configurations. Future improvements can focus on modifying this approach to a wider range of diverse and constantly evolving scenarios, improving the algorithm's usefulness and resilience. Although the proposed method has shown encouraging outcomes in offline path planning and obstacle avoidance, further research could investigate the incorporation of dynamic obstacle detection and avoidance capabilities. This would allow the algorithm to adjust to rapidly changing environments while executing a mission. Moreover, extending the approach to three-dimensional path planning and including additional constraints, such as energy efficiency and environmental considerations, could further enhance its applicability in various operational scenarios. This research provides a substantial contribution to the field of UAV path planning by introducing an innovative and effective solution that resolves the key problems of optimising missions, avoiding obstacles, and ensuring operational safety in applications like WSN and precision agriculture. The proposed integrated method not only enhances the theoretical underpinnings of path planning algorithms but also positions itself as a promising option for enabling efficient, secure, and autonomous UAV operations in numerous real-world applications.

Author Contributions: Conceptualisation, D.D., F.V., S.B. and F.G.; methodology, D.D., S.B., F.V. and F.G.; hardware conceptualisation and integration, D.D., S.B. and F.V.; software, D.D. and F.V; validation, D.D., F.V. and F.G.; formal analysis, D.D.; investigation, D.D. and F.V.; resources, F.G.; data curation, D.D.; writing—original draft preparation, D.D.; writing—review and editing, F.V., S.B. and F.G.; visualisation, D.D.; supervision, F.V. and F.G.; project administration, F.G.; funding acquisition, F.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors acknowledge continued support from the Queensland University of Technology (QUT) through the QUT Centre for Robotics (QCR) and Engineering Faculty for allowing access and flight tests at O block, QUT.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Sandino, J.; Galvez–Serna, J.; Mandel, N.; Vanegas, F.; Gonzalez, F. Autonomous mapping of desiccation cracks via a probabilistic-based motion planner onboard uavs. In Proceedings of the 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 5–12 March 2022; pp. 1–14.
2. BinKai, Q.; Mingqiu, L.; Yang, Y.; XiYang, W. Research on UAV path planning obstacle avoidance algorithm based on improved artificial potential field method. *Proc. J. Phys. Conf. Ser.* **2021**, *1948*, 012060. [CrossRef]
3. Debnath, D.; Hawary, A.F.; Ramdan, M.I.; Alvarez, F.V.; Gonzalez, F. QuickNav: An Effective Collision Avoidance and Path-Planning Algorithm for UAS. *Drones* **2023**, *7*, 678. [CrossRef]
4. Zhang, Z.; Liu, X.; Feng, B. Research on obstacle avoidance path planning of UAV in complex environments based on improved Bézier curve. *Sci. Rep.* **2023**, *13*, 16453. [CrossRef] [PubMed]
5. Hayat, S.; Yanmaz, E.; Muzaffar, R. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2624–2661. [CrossRef]
6. Jiang, B.; Bishop, A.N.; Anderson, B.D.; Drake, S.P. Optimal path planning and sensor placement for mobile target detection. *Automatica* **2015**, *60*, 127–139. [CrossRef]
7. Sandino, J.; Maire, F.; Caccetta, P.; Sanderson, C.; Gonzalez, F. Drone-based autonomous motion planning system for outdoor environments under object detection uncertainty. *Remote Sens.* **2021**, *13*, 4481. [CrossRef]
8. Sung, I.; Nielsen, P. Zoning a service area of unmanned aerial vehicles for package delivery services. *J. Intell. Robot. Syst.* **2020**, *97*, 719–731. [CrossRef]
9. Babel, L. Coordinated target assignment and UAV path planning with timing constraints. *J. Intell. Robot. Syst.* **2019**, *94*, 857–869. [CrossRef]
10. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A survey of path planning algorithms for mobile robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]
11. Rossi, M.; Brunelli, D. Autonomous gas detection and mapping with unmanned aerial vehicles. *IEEE Trans. Instrum. Meas.* **2015**, *65*, 765–775. [CrossRef]
12. Primatesta, S.; Rizzo, A.; la Cour-Harbo, A. Ground risk map for unmanned aircraft in urban environments. *J. Intell. Robot. Syst.* **2020**, *97*, 489–509. [CrossRef]
13. Xu, Y.; Che, C. A brief review of the intelligent algorithm for traveling salesman problem in UAV route planning. In Proceedings of the 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 12–14 July 2019; pp. 1–7.
14. Cakir, M. 2D path planning of UAVs with genetic algorithm in a constrained environment. In Proceedings of the 2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), Istanbul, Turkey, 27–29 May 2015; pp. 1–5.
15. Popescu, D.; Stoican, F.; Stamatescu, G.; Chenaru, O.; Ichim, L. A survey of collaborative UAV–WSN systems for efficient monitoring. *Sensors* **2019**, *19*, 4690. [CrossRef] [PubMed]
16. Augello, A.; Gaglio, S.; Lo Re, G.; Peri, D. Time-Constrained Node Visit Planning for Collaborative UAV–WSN Distributed Applications. *Sensors* **2022**, *22*, 5298. [CrossRef] [PubMed]
17. Karegar, P.A.; Al-Anbuky, A. Travel path planning for UAV as a data collector for a sparse WSN. In Proceedings of the 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), Pafos, Cyprus, 14–16 July 2021; pp. 359–366.
18. Zhu, Y.; Wang, S. Efficient aerial data collection with cooperative trajectory planning for large-scale wireless sensor networks. *IEEE Trans. Commun.* **2021**, *70*, 433–444. [CrossRef]
19. Tsouros, D.C.; Bibi, S.; Sarigiannidis, P.G. A review on UAV-based applications for precision agriculture. *Information* **2019**, *10*, 349. [CrossRef]
20. Srivastava, K.; Pandey, P.C.; Sharma, J.K. An approach for route optimization in applications of precision agriculture using UAVs. *Drones* **2020**, *4*, 58. [CrossRef]
21. Liu, J. An improved genetic algorithm for rapid uav path planning. *Proc. J. Phys. Conf. Ser.* **2022**, *2216*, 012035. [CrossRef]
22. Rafai, A.N.A.; Adzhar, N.; Jaini, N.I. A review on path planning and obstacle avoidance algorithms for autonomous mobile robots. *J. Robot.* **2022**, *2022*, 2538220. [CrossRef]
23. Xin, J.; Zhong, J.; Yang, F.; Cui, Y.; Sheng, J. An improved genetic algorithm for path-planning of unmanned surface vehicle. *Sensors* **2019**, *19*, 2640. [CrossRef]
24. Roberge, V.; Tarbouchi, M.; Labonté, G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Trans. Ind. Inform.* **2012**, *9*, 132–141. [CrossRef]
25. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]
26. Rao, Y.; Cao, J.; Zeng, Z.; Duan, C.; Wei, X. An Improved Ant Colony Algorithm for UAV Path Planning in Uncertain Environment. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–10.
27. Yang, F.; Fang, X.; Gao, F.; Zhou, X.; Li, H.; Jin, H.; Song, Y. Obstacle avoidance path planning for UAV based on improved RRT algorithm. *Discret. Dyn. Nat. Soc.* **2022**, *2022*, 4544499. [CrossRef]

28. Lin, H.-Y.; Peng, X.-Z. Autonomous quadrotor navigation with vision based obstacle avoidance and path planning. *IEEE Access* **2021**, *9*, 102450–102459. [CrossRef]
29. Pehlivanoglu, Y.V.; Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Appl. Soft Comput.* **2021**, *112*, 107796. [CrossRef]
30. Hu, J.; Yang, X.; Wang, W.; Wei, P.; Ying, L.; Liu, Y. Obstacle avoidance for uas in continuous action space using deep reinforcement learning. *IEEE Access* **2022**, *10*, 90623–90634. [CrossRef]
31. AlRaslan, M.; AlKurdi, A.H. UAV Path Planning using Genetic Algorithm with Parallel Implementation. *Int. J. Comput. Sci. Inf. Technol.* **2022**, *10*, 1–15. [CrossRef]
32. Chen, J.; Du, C.; Han, P.; Zhang, Y. Sensitivity analysis of strictly periodic tasks in multi-core real-time systems. *IEEE Access* **2019**, *7*, 135005–135022. [CrossRef]
33. Razzaghi, P.; Tabrizian, A.; Guo, W.; Chen, S.; Taye, A.; Thompson, E.; Bregeon, A.; Baheri, A.; Wei, P. A survey on reinforcement learning in aviation applications. *arXiv* **2022**, arXiv:2211.02147.
34. Fu, Z.; Yu, J.; Xie, G.; Chen, Y.; Mao, Y. A heuristic evolutionary algorithm of UAV path planning. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 2851964. [CrossRef]
35. Purkayastha, R.; Chakraborty, T.; Saha, A.; Mukhopadhyay, D. Study and analysis of various heuristic algorithms for solving travelling salesman problem—A survey. In Proceedings of the Global AI Congress 2019, Kolkata, India, 12–14 September 2019; Advances in Intelligent Systems and Computing. Springer: Singapore, 2020; Volume 1112, pp. 61–70.
36. Silva Arantes, J.d.; Silva Arantes, M.d.; Motta Toledo, C.F.; Júnior, O.T.; Williams, B.C. Heuristic and genetic algorithm approaches for UAV path planning under critical situation. *Int. J. Artif. Intell. Tools* **2017**, *26*, 1760008. [CrossRef]
37. Chen, J.; Li, M.; Yuan, Z.; Gu, Q. An improved A* algorithm for UAV path planning problems. In Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 12–14 June 2020; pp. 958–962.
38. Zhang, J.; Liu, Z.; Wang, Y.; Zhang, F.; Li, Y. Research on effective path planning algorithm based on improved A* algorithm. *Proc. J. Phys. Conf. Ser.* **2022**, *2188*, 012014. [CrossRef]
39. TSPLIB. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ (accessed on 9 May 2024).
40. Boiteau, S.; Vanegas, F.; Gonzalez, F. Framework for Autonomous UAV Navigation and Target Detection in Global-Navigation-Satellite-System-Denied and Visually Degraded Environments. *Remote Sens.* **2024**, *16*, 471. [CrossRef]