Applied Soft Computing Journal 90 (2020) 106168

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Implementation of optimized dynamic trajectory modification algorithm to avoid obstacles for secure navigation of UAV

P.S. Krishnan [a], K. Manimala [b,*]

[a] Aeronautics Research and Development Board, DRDO, Bangalore, India
[b] Dr. Sivanthi Aditanar College of Engineering, Tiruchendur, Tamilnadu, India

ARTICLE INFO

Article history:
Received 11 January 2019
Received in revised form 6 November 2019
Accepted 8 February 2020
Available online 20 February 2020

Keywords:
Collision avoidance
Six Degrees of Freedom (6-DOF)
Obstacle detection
Particle Swarm Optimization (PSO)
Unmanned Aerial Vehicle (UAV)

ABSTRACT

To develop escape manoeuvre from obstacles and to find new waypoints for dynamic trajectory modification of UAV, a novel Particle Swarm Optimization based Collision Avoidance algorithm (PSO-CA) is presented in this paper. The proposed "obstacle sense and avoid" algorithm and the logical decision-making system aids the Unmanned Aerial Vehicle (UAV) to re-route its current path to a safer flight course when an obstacle pops up along its intended path. A radar with 10 km range identifies obstacles and the UAV manoeuvres based on radar data, making it suitable for any unknown environment. The proposed system would manoeuvre the UAV autonomously along optimized alternate path to avoid the conflicting traffic. New waypoints are identified and the waypoint list is modified dynamically to avoid collision with stationary threats like hill, tree and moving intruders like other UAVs. The proposed algorithm steers the vehicle safely along alternate path with less change in intended trajectory while avoiding all potential threats. As the PSO-CA algorithm detects obstacles and identifies alternate path well in advance for unknown pop up threats, the UAV is safe and is suitable for real time environment. The proposed algorithm has considered obstacles with different positions, different sizes and random motion. Experimental results conducted on 6-DOF model UAV with different obstacles clearly justify the efficiency of the proposed method in comparison with other planners.

1. Introduction

There is a necessity for military leaders and civilian visionaries [1] to request that UAVs be allowed access into international flight routes in order to capitalize on their performance and expand their area of operation. However, limitations in areas such as reliable obstacle detection and collision avoidance systems have restricted the use of UAVs to ground control and human intervention in course planning and control, in addition to the confinement of this type of aircraft to segregated airspace. Over the past few decades, autonomous path planning techniques for UAV are becoming significant, because the conventional remotely piloted techniques could not provide either satisfactory accuracy or perfect timing for military or civilian missions. As the technology advances enable autonomous long endurance missions to be conducted using UAVs, the human factor of ground control must be eliminated. Thus, the UAVs must have a "Detect, Sense and Avoid" system that allows detecting and safely changing paths in order to avoid the obstacles along its trajectory. Real time dynamic trajectory revision is required whenever the UAV faces sudden threat.

1.1. Existing research

To show the significance of UAV collision avoidance & path planning, a few recent literature works are reported. In [2], Naifeng Wen modelled Static threats (STs) using Intuitionistic Fuzzy Set (IFS) and predicted Dynamic Threats (DT) using reachability set (RS). By improving dynamic domain rapidly-exploring random tree (DDRRT), UAV path planner was designed for dealing with complex obstacles. Peng Yang et al. [3] proposed idea of separately evaluating and evolving waypoints. Waypoints on UAV's path were evaluated separately by a set of new evaluation functions. In [4], the authors proposed the design and implementation of sampling-based path planning methods for UAV to avoid collision with commercial aircraft and other moving obstacles. For path planning of unmanned aerial vehicle (UAV) Yangguang Fu et al. [5], suggested hybrid Differential Evolution (DE) with quantum-behaved particle swarm optimization (DEQPSO). In DE-QPSO, the donor vector is generated by adding the weighted difference between two or more personal best positions to the global best position obtained by QPSO. The differential operator in DEQPSO is done on the best positions and not on the individuals of the current population.

Several methodologies are proposed to solve this complex optimization problem, such as potential field based methods

* Corresponding author.
E-mail address: s_monimala@yahoo.com (K. Manimala).

https://doi.org/10.1016/j.asoc.2020.106168
1568-4946/

(PFM) [6], A* algorithm [7], homotopy continuation method [8], heuristic approaches [9], and so on. Finding the optimal path for autonomous UAV is an NP-complete problem [10], and the complexity increases as the size of the problem increases. Literature reports several bio-inspired techniques for solving UAV path planning [11]. These include genetic algorithm (GA) [10,12], particle swarm optimization (PSO) [12–14], ant colony optimization (ACO) [15], artificial bee colony (ABC) algorithm [16], central force optimization (CFO) [17], differential evolution (DE) algorithm [18], bat algorithm (BA) [11], imperialist competitive algorithm (ICA) [19,20], fruit fly optimization algorithm [21] and the hybrid versions of the above algorithms [22,23]. Collision avoidance methods are suggested in the works reported by literatures [24–32] and evolutionary methods for path planning is described in the works [33–36]. Even though several works report about collision avoidance and trajectory modification [3,36,37], optimization process is not explored by them [26,27].

1.2. Research gap

Despite the availability of enormous literature, there are many challenges that needs attention in the area of UAV path planning. The research gap includes (1) Path planning algorithm is mostly done as off line analysis considering the terrain data and prior fixed stationary obstacles. Many of the existing works have not explored online dynamic trajectory revision when sudden unknown threats pop-up along its path (2) The features of the flying space and threats are set in advance, including geographical range, topographical conditions, danger zones, radar exposure area, threat locations, threat levels, and so on. The existing path planning literatures have not reported for collision avoidance in an unknown environment (2) for practical implementation of the collision avoidance algorithm in real flights, the dynamic constraints like pitch rate, banking angle and radius of curvature should be strictly adhered. The newly formed trajectory after collision avoidance need to keep the control inputs and system states within allowable limits and (3) the heuristic algorithms employed should not be trapped into a local minimum. (4) the new trajectory may not be the best path due to algorithmic constraints. Inspired by the above-mentioned factors, this study develops an optimal path planner to avoid stationary and moving threats using Particle Swarm Optimization (PSO) based Collision Avoidance algorithm (PSO-CA).

1.3. Novelty of the proposed work

The main contribution of this paper is that it develops a novel PSO based collision avoidance algorithm to solve the UAV path modification problem, which can be suited in complex unknown environments. This work differs from existing works in many ways. This paper focuses on online alternate path planning when a radar mounted UAV flying along designated trajectory faces sudden unknown threats along its designated path thus facilitating autonomous navigation. Several existing works focused on fixed obstacles whereas this work reports collision avoidance scheme for random obstacles that comes under view only when those obstacles falls within radar range. The novelty of this work includes

- A novel optimized path planning algorithm is designed to avoid stationary and moving obstacles while traversing the designated path from source to destination.
- Path planning for three-dimensional dynamic real-world environment with obstacles of various dimensions, random position and random motion
- The path planner takes into consideration the dynamics and constraints of UAV, so that the newly identified alternate trajectory is feasible for the UAV
- Identification of alternate waypoints to avoid collision
- Obstacles include sudden pop-up stationary and moving threats captured by radar unit of UAV
- Collision avoidance with voluminous obstacles like hills based on radar data
- Considering multiple collisions including stationary and moving obstacles
- UAV manoeuvres based on radar data and no prior information of terrain data is required making the UAV to fly in any unknown environment
- Earlier detection of obstacles (when it is captured by radar with 10 km range) and finding alternate optimized path well in advance ensuring safety of UAV

The remaining section of this paper is structured as follows. Problem formulation is given in Section 2. The frame work of the proposed system is detailed in Section 3. The proposed PSO algorithm and parameter setting is described in Section 4. Results are discussed in Section 5 and the comparison with existing works done in Section 6. Finally, the paper is concluded in Section 7.

2. Problem formulation

2.1. Problem definition

In path planning, UAV has to detect the presence of obstacles along its predetermined trajectory through designated waypoints, avoid them by identifying new waypoints for alternate path with less travel distance and maximum safety

2.1.1. UAV model & mission

UAV is modelled using 6-DOF equations and navigated along a random set of initial waypoints. The initial waypoints are chosen in random which includes straight line path as well as turning movements. This work demonstrates the performance of PSO-CA algorithm for a random set of 8 waypoints with straight line path between any 2 waypoints. For a UAV mission with 8 waypoints as shown in Fig. 1, the UAV has to travel along 7 segments or legs. Leg represents the distance between two waypoints. Let us assume the UAV is flying at altitude z during the mission. WP1=$\{x_1, y_1, z\}$ and WP2=$\{x_2, y_2, z\}$ represents the 3-dimensional waypoint positions. UAV starts from WP1 and travels along leg1 to reach WP2. The start and destination points are $X_{st}=x_1$; $Y_{st}=y_1$; $X_{fin}=x_2$; $Y_{fin}=y_2$. The UAV changes its head before reaching WP2 (<1 km) and starts heading towards WP3. i.e when the distance to reach WP2 is less than 1 km, the UAV changes its head towards WP3. The flight controller module receives the command $X_{st}=x_2$; $Y_{st}=y_2$; $X_{fin}=x_3$; $Y_{fin}=y_3$. This is how the leg change over takes place. The path planning algorithm in this paper is explained by the mission of travel from WP1 to WP2 as shown in Fig. 1(b). The obstacles beyond WP2 are considered in the next leg of travel from WP2 to WP3.

A 6-DOF solar UAV was designed in Matlab software and allowed to navigate in the predetermined trajectory along waypoints given in the waypoint list. The UAV travels at a speed of 15 m/s. The control commands, fixed parameters of UAV and other equations for designing 6-DOF UAV are taken from [38–42]. The values of certain parameters of 6-DOF model are given in Appendix. An outline of the 6-DOF model representation is shown in Fig. 2.

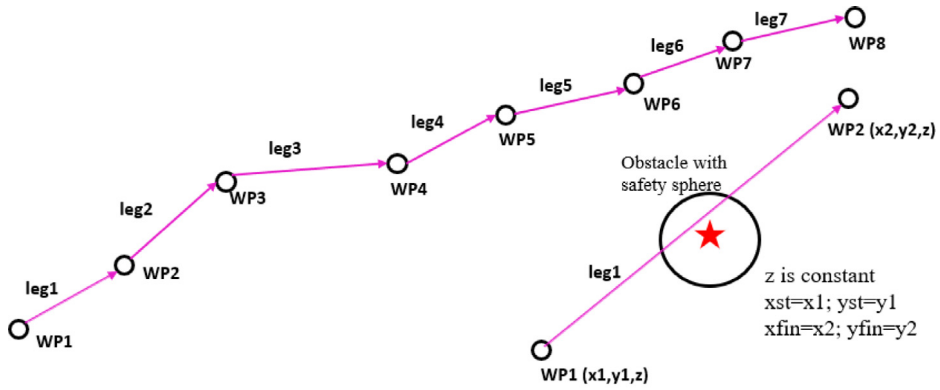


Fig. 1. UAV mission.

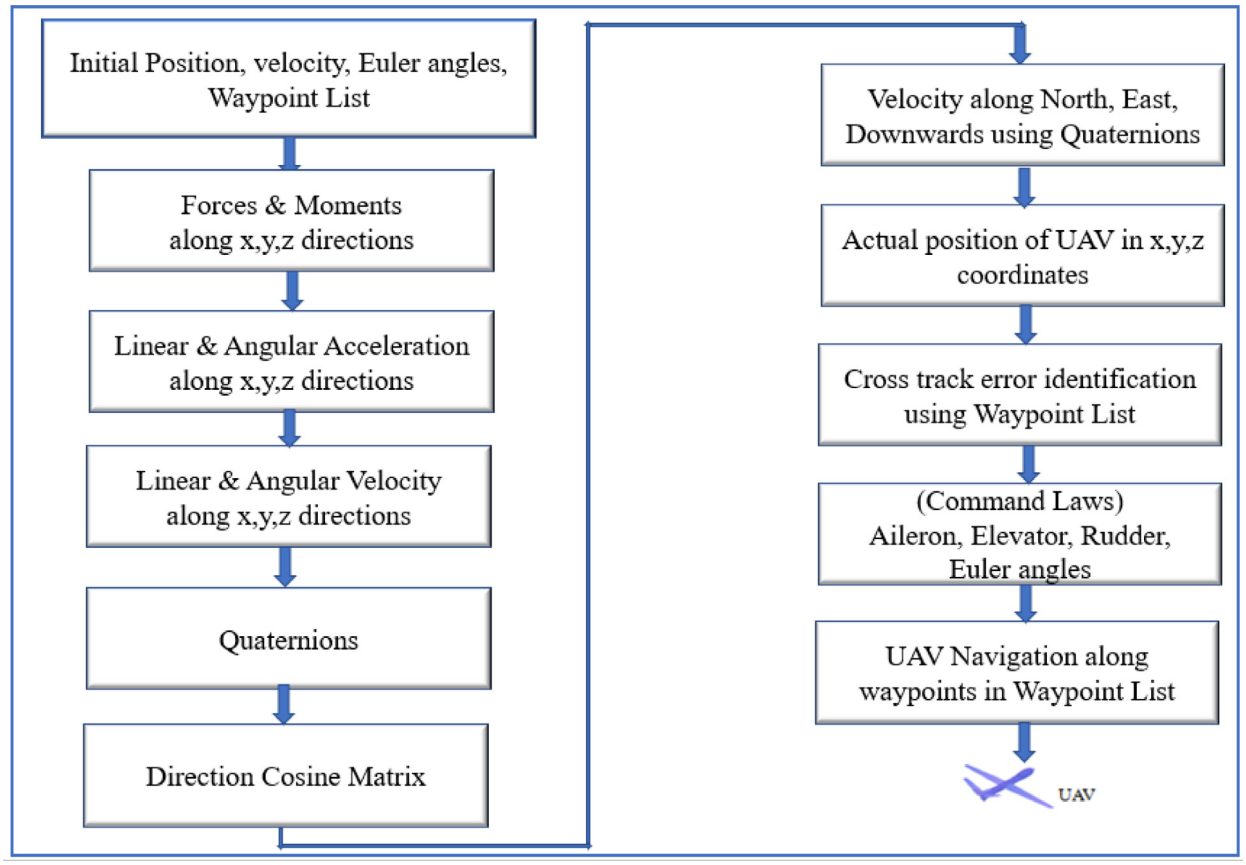


Fig. 2. 6-DOF UAV Model.

2.1.2. Obstacle model

Obstacles are modelled as in Fig. 3. A single stationary obstacle is modelled as point (x_0, y_0, z) with safety sphere of radius *RS*. Multiple obstacles are modelled as scattered stationary obstacles. Voluminous stationary obstacle like mountain is modelled using group of single obstacles with little distance between them. The terminal obstacles with safety sphere around them are identified for avoidance. The moving obstacles are modelled as 6-DOF UAV.

2.1.3. Problem description

The obstacle avoidance problem is discussed by positioning obstacles along the path of travel in leg from one waypoint to another. Collision is detected when the straight-line trajectory of UAV intersects the safety sphere of obstacle. The radar range for detecting obstacle is 10 km and the algorithm is designed to avoid obstacles between two waypoints. The collision avoidance system should identify alternate path with less travel distance, maximum safety and minimum deviation from intended trajectory.

If there is a single obstacle between two waypoints as in Fig. 1(b), the algorithm could easily find a new waypoint along the shortest chord of the safety sphere to ensure short travel distance. But if there is a presence of another one obstacle along the shortest chord of safety sphere, then the path planner has to choose a waypoint along the long chord of the safety sphere to ensure safety. If some more obstacles are present along both sides of the safety sphere of obstacle being considered, then the

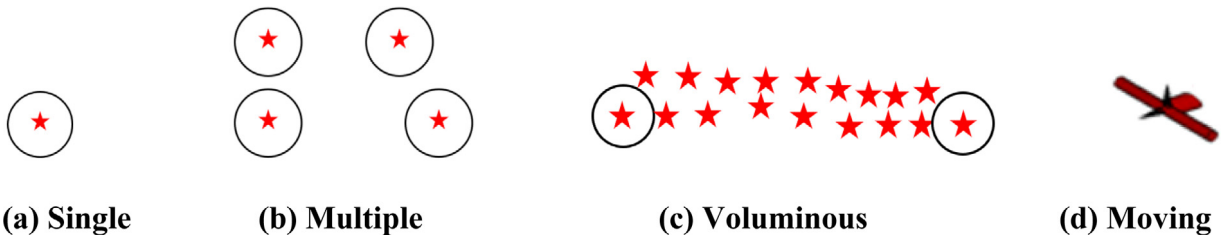


Fig. 3. Obstacle Modelling.

path planning algorithm has to search for a feasible path without collision which explains the complexity of the problem. As the number of obstacles increases, the path planner finds it difficult to obtain feasible path as the space for UAV movement decreases rapidly. Also, the increase in obstacles makes the pathway much narrower and zigzagged. Hence, more waypoints are required to make the path flyable, i.e., smooth and safe, and thus, a suitable path planning algorithm is required to identify the optimized number and location of new waypoints.

2.2. Need for optimization

The path-planning problem for the safe journey of UAV can be formulated as an optimization problem that searches a realistic path from the start point to the destination [43]. As per literature, a path is a set of segments through a sequence of waypoints. The segments used are line segments [43], Bspline curves or Bezier curves [37]. Hence, the dynamic path planning problem identifies sequence of waypoints and the segments connecting each pair of adjacent waypoints to optimize diverse objectives subject to several constraints. The curve-based representations can ensure smoothness of candidate paths, but their computational costs are high as they need external local controls for generating paths. In this paper, line-segment-based paths are used because of their simplicity and efficiency.

When designing a collision avoidance system for UAV, few significant aspects should be considered, such as the manoeuvrability of the UAV, the surrounding environment of the mission, the constraints of UAV, safety, and cost of the path. These aspects are incorporated in the form of objective functions that need to be minimized, and in the form of constraints that a path must act in accordance with. The first aspect is the location of new waypoints and the distance taken to reach it along with the proximity of other obstacles. The second aspect is relevant to the segments or legs (path between current position of UAV and the new waypoint chosen) as well as to the waypoints, since waypoints are not sufficient to determine the real states of UAV. In other words, the segments may be infeasible even though the corresponding waypoints are in feasible locations [3].

2.3. Optimization problem formulation

The design variables or parameters are taken as $p = \{x_p, y_p\}$ where x_p and y_p represents the x and y position of new waypoint.

The optimization problem is set as

$$\boldsymbol{minimize_p\ f\ (x_p, y_p)}$$

$$\boldsymbol{subject\ to\ g\ (x_p, y_p)} \geq 0 \tag{1}$$

The problem is to minimize $f\ (x_p, y_p)$ by choice of x_p and y_p keeping altitude z constant. Since the climbing speed of the UAV considered is very less (3 m/s), the altitude change manoeuvre is not taken into account and hence z is not changed. The function $g\ (x_p, y_p) \geq 0$ represents the inequality constraints.

2.3.1. Objective functions

2.3.1.1. Minimal path length. For military missions, shorter paths are always preferred to longer ones, because shorter paths consume less fuel and have less chance of encountering unexpected threats, e.g., undetected enemies. Hence, the total length of the path needs to be minimized. This consideration leads to the objective function path length f_1

$$f_1 = ((x_p - x_{pos})^2 + (y_p - y_{pos})^2)^{1/2} + ((x_{fin} - x_p)^2 + (y_{fin} - y_p)^2)^{1/2} \tag{2}$$

(x_{pos}, y_{pos}) represents the current x and y position of UAV, (x_p, y_p) represents the position of new waypoint and (x_{fin}, y_{fin}) represents the position of next way point or the destination to be reached in this leg.

2.3.1.2. Minimal risk of other obstacles. If the new waypoint is within the range of other obstacles, it is at risk. Intuitively, paths with lower risk of other obstacles are safer than those with higher probabilities. Presence of other obstacles ($k = 1,2\ldots M$) impose a certain risk of collision for the UAV only if that point is inside the region defined by the other obstacle's maximal risk distance i.e *MXRSK* of the obstacle.

The distance between a new waypoint (x_p, y_p) and the kth obstacle (x_o^k, y_o^k) is calculated as

$$dis^k = ((x_p - x_o^k)^2 + (y_p - y_o^k)^2)^{1/2} \tag{3}$$

$$\text{Risk of collision if } dis^k < MXRSK^k \tag{4}$$

The total number of obstacles N_0, which imposes risk of collision with the new waypoint is identified based on the above condition

This consideration leads to the objective function risk f_2

$$f_2 = \left(\sum_{k=1}^{N_0} (MXRSK^k - dis^k)\right)^2 \tag{5}$$

This objective function is designed to penalize waypoints near obstacles.

2.3.1.3. Selection of the final solution. The proposed collision avoidance problem is multiobjective optimization problem (MOP) at first sight as there are two conflicting objective functions to optimize. Usually, a common practice in the context of UAVs is to integrate different objectives by using weighted sum [34–36]. The two objectives are combined using weights to obtain final objective function f as

$$f = W_t f_1 + W_0 f_2 \tag{6}$$

W_t represents weighting factor for travel path, W_o represents weighting factor to penalize the solution which has more other dangers (obstacles) nearby. $W_t + W_o = 1$. W_t is set as 0.1 and W_o as 0.9. The solution which has less obstacle nearby should be preferred and hence more weight is assigned for risk of obstacles than for travel distance. This optimization is formulated to give more priority for safety than for travel path length.

2.3.2. Constraints function

2.3.2.1. Safe waypoint. The new waypoint (x_p, y_p, z) identified should be at a distance *RS* away from obstacle (x_o, y_0, z) to avoid collision.

$$g_1 = \left(\left(x_p - x_o\right)^2 + \left(y_p - y_o\right)^2\right)^{1/2} > RS + tolerance \tag{7}$$

2.3.2.2. Search space limit. The UAV should not deviate much away from the intended trajectory while avoiding collision. Hence, the new waypoint identified should be within *searchspaceLimit*.

$$g_2 = \left(\left(x_p - x_o\right)^2 + \left(y_p - y_o\right)^2\right)^{1/2} < searchspaceLimit \tag{8}$$

2.3.2.3. Feasible segment. The waypoints are not sufficient to determine the real states of UAV. In other words, the segments may be infeasible even when the corresponding waypoints are in feasible locations. The segment through the new waypoint (x_p, y_p) to the next waypoint (x_{fin}, y_{fin}) should not pass through the safety sphere of obstacle being avoided.

$$g_3 = b^2 - 4ac < 0 \tag{9}$$

where, $a = (x_{fin} - x_p)^2 + (y_{fin} - y_p)^2 + (z_{fin} - z_p)^2$

$$b = 2\left[\left(x_{fin} - x_p\right)\left(x_p - x_o\right) + \left(y_{fin} - y_p\right)\left(y_p - y_o\right) + (z_{fin} - z_p)(z_p - z_o)\right]$$

$$c = x_o^2 + y_o^2 + z_o^2 + x_p^2 + y_p^2 + z_p^2 - 2\left(x_o x_p + y_o y_p + z_o z_p\right) - RS^2 \tag{10}$$

2.3.2.4. Maximal turning angle. The radius of turn of UAV is limited by the current velocity and maximal banking angle. The waypoint which requires less radius of turn to reach it could not be accepted. The angle between its previous direction and the current direction in the horizontal direction is termed as turning angle. This work uses the formulation for turning angle as suggested by Zheng et al. [43].

$$g_4 = \theta_i = \arccos \times \left(\frac{(x_i - x_{i-1}, y_i - y_{i-1}) \,.\, (x_{i+1} - x_i, y_{i+1} - y_i)^T}{\|(x_i - x_{i-1}, y_i - y_{i-1}) \,.\, (x_{i+1} - x_i, y_{i+1} - y_i)\|}\right) \tag{11}$$

where $\|x\|$ means the norm of vector *x*.

2.3.2.5. Design of constraints function.

$$g\left(x_p, y_p\right) = \left\{\begin{array}{c} g_1 > RS + tolerance \\ g_2 < searchspaceLimit \\ g_3 < 0 \\ g4 \leq MaxBankAngle \end{array}\right\} \tag{12}$$

RS represents Radius of safety sphere (1 km in this work) around obstacle and *tolerance* an additional distance to ensure maximum safety (1 km in this work), *searchspaceLimit* is the limit to restrict the search space distance for new waypoint (4 km). The maximum Bank angle *MaxBankAngle* for the UAV designed is 30°.

There are 4 inequality constraints in this optimization problem. The first two constraints are for finding new waypoints. While the first constraint avoids collision with obstacle, the second constraint strictly restricts search space within particular limit thereby restricting UAV nearby the intended trajectory. The third constraint is for feasible segment to the new waypoint. The fourth one deals with the vehicle's turning constraint.

2.4. PSO for optimization of waypoints

A candidate solution to the path planning problem is encoded as a real valued two-dimensional vector that represents the positions of waypoints (x and y position), and the optimal path is iteratively searched in the corresponding real space. A number of candidate paths are generated in each iteration and evaluated with respect to the objective functions and constraints. To solve this optimization problem of collision avoidance, Particle Swarm Optimization technique (PSO) is chosen for the following reasons

(i) Collision avoidance is a real time issue and has strict time constraint
(ii) PSO is simple and takes less time for computation
(iii) PSO starts with population of solution and searches the entire search space
(iv) No genetic operations like mutation and crossover in PSO
(v) Converges quickly

3. Framework of proposed system

3.1. Architecture of proposed system

The functional block diagram of the proposed system is shown in Fig. 4. The UAV's flight controller obtains predefined waypoints list and issue necessary control commands to navigate the UAV along predefined trajectory. Upon receiving obstacle data from radar module, the collision detection module detects the collision possibility. The collision avoidance module identifies initial solution or particles (population size of 10) away from danger area of obstacle and activates PSO algorithm to find the optimized new waypoint. The original waypoint list is now updated after including the optimized new waypoint and passed on to the flight controller module for safer course of action.

The proposed system consists of 6 modules namely the Main Module, Radar Module, Flight Controller Module, Cross track error avoidance module, Collision Detection and Collision Avoidance Module.

3.2. List of symbols and variables

The list of symbols and variables used in this paper are listed in Table 1.

3.3. Main module

The main controller module is responsible for the overall operation of the flight system. It interacts with all other modules in the system. It receives the list of waypoints to travel along and initiates necessary control through the flight controller module for UAV navigation. It would continuously "ping" or request information from the radar unit. The radar module simulates a list of several stationary and moving obstacles and they are observed when they comes under the radar range of 10 km. The radar module also adds up several popped-up threats which are induced at certain period of time. The radar in the UAV provides the details of obstacles located in the scan zone for every 1 s, in the form of an array which contains unique identification number (ID) and position to the main module. The main program detects the position of obstacle between waypoints and discriminates whether it is stationary or moving. It also detects continuous obstacles like mountain or hill and captures the edges of it to identify alternate path. The collision detection module predicts collision once the obstacle falls under radar view and activates collision-avoidance module for necessary action.

The main module of the algorithm has the following responsibilities:

- Initialization of all flight parameters and waypoints to fly
- Controller of all modules namely radar module, flight controller module, collision detection and avoidance module

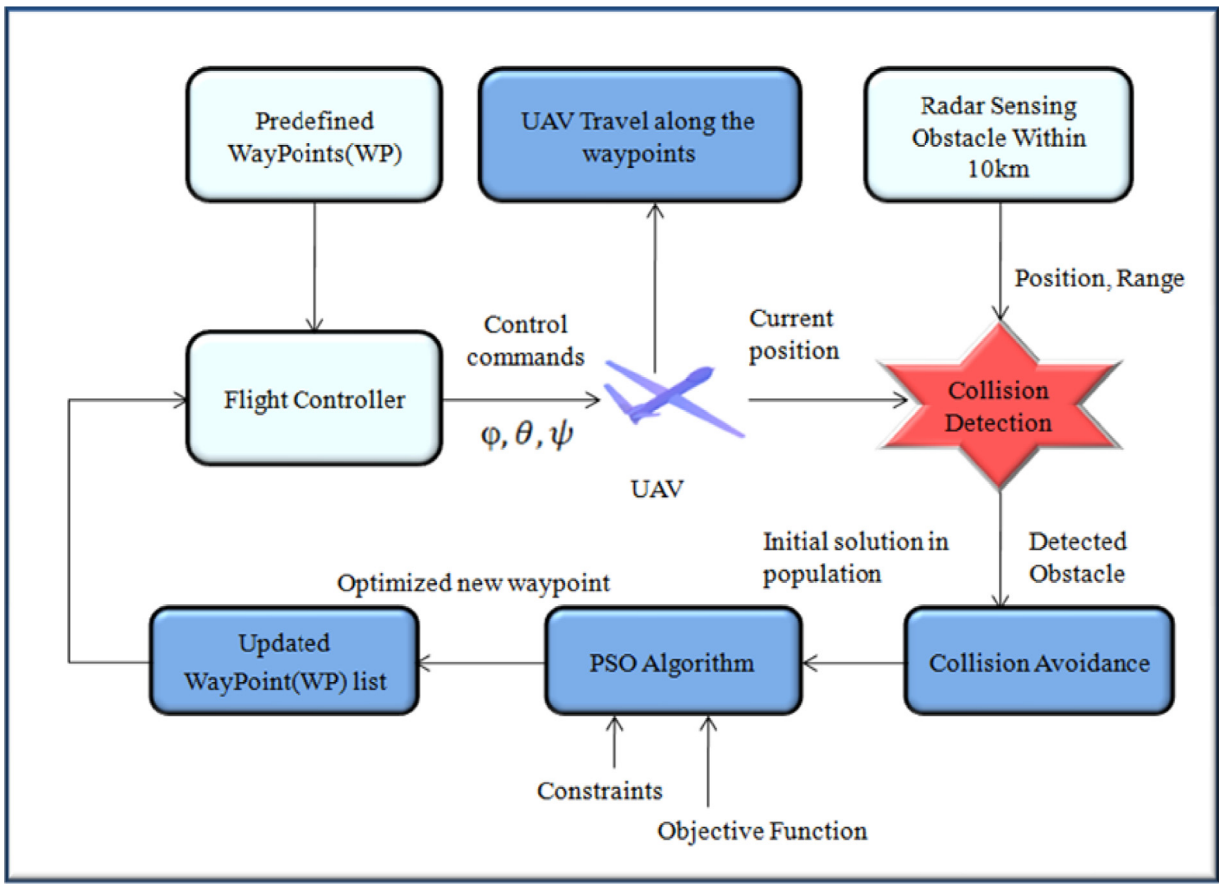


Fig. 4. Functional Block Diagram.

Table 1
List of symbols and variables.

Symbol or variable	Explanation
UAV	Unmanned Aerial Vehicle
6-DOF	Six Degree of Freedom
$WP = \{WP_1, WP_2 \ldots..WP_8\}$	Waypoint list containing all the waypoints along UAV's predetermined trajectory
WP1, WP2	Waypoint 1 and Waypoint2
$\{x_1, y_1, z\}, \{x_2, y_2, z\}$	x, y and z coordinates of waypoint1 and waypoint2
leg	Distance between two waypoints
$(X_{st}, Y_{st}), (X_{fin}, Y_{fin})$	Starting and final waypoints of travel along leg
RS	Radius of safety sphere around obstacle
p	Design variable or parameter of optimization
$\{x_p, y_p\}$	x and y position of optimized new waypoint.
(x_{pos}, y_{pos})	current x and y position of UAV
(x_o, y_o)	x and y position of obstacle
N_0	Total number of obstacles
MXRSK	maximal risk distance
W_t	weighting factor for travel path
W_o	weighting factor to penalize solution near obstacles
tolerance	extra safety distance around obstacle (1 km)
searchSpaceLimit	Maximum limit for searching optimized solution (4 km)
MaxBankAngle	Maximum Bank angle (30 degrees)
alt	Altitude or z coordinate = 200 m
iter, maxiter	Current iteration and maximum iteration
φ, θ, Ψ	Euler angles (Bank, Elevation, Heading angles)
DD	Cross track error
CD	Collision Distance
pbest	Best solution in each iteration for PSO algorithm
gbest	Global best solution for PSO algorithm
I, *k*,	Total iterations, current iteration number
V^k	Velocity in current iteration of PSO algorithm
X^k	Current position of UAV

- Identify whether the obstacle is stationary or moving
- Identify whether single or finite volume obstacle

1. *procedure main*
2. Initialize Flight parameters, waypoints $WP = \{WP_1, WP_2 \ldots\ldots WP_8\}$, radius of safety sphere *RS* around obstacle
3. *legtotal*=waypoints-1
4. $leg = 1$, $X_{st} = WP(leg), X_{fin} = WP(leg + 1)$
5. **while (***leg* ≤ *legtotal)*
6. UAV moves along *leg* from $X_{st}\ to\ X_{fin}$
7. Radar in UAV scans for obstacles within 10 km
8. **if** obstacle detected **then**
9. do *collision_possibility* for all detected obstacles
10. **if** collision **then** do *stationary_moving* **else** go to step 26
11. **if** moving go to step 20
12. **else if** stationary **then** do *continuous obstacle*
13. **if** obstacle is single **then** do *near_ end _waypoint*
14. **if** *near_ end _waypoint* skip end waypoint and do *Find_InitialNew_Waypoints* go to step 23
15. **else if** *obs_bet_wps* **then** do *Find_InitialNew_Waypoints*, go to step 23
16. **end if**
17. **else if** *continuous_obstacle* **then** find start and end point of obstacles and do
18. *Find_InitialNew_Waypoints*, go to step 23
19. **end if**
20. **do** *confirm_collision* and do *Find_InitialNew_Waypoints* go to step 23
21. **end if**
22. **end if**
23. do *PSO_optimized _waypoint*
24. $X_{fin} = optimized\ waypoint$ & update waypoint list
25. **end if**
26. Navigate UAV based on waypoint list
27. To smooth UAV's path during leg changeover
if distance between X_{fin} and UAV position < 1 Km, **then**
28. $X_{st} = X_{fin}\ and\ X_{fin} = WP(leg + 1)$
29. **end if**
30. Calculate and correct cross track error
31. *leg* = *leg* + 1
32. end *procedure*

Fig. 5. Algorithm 1: m*ain module*.

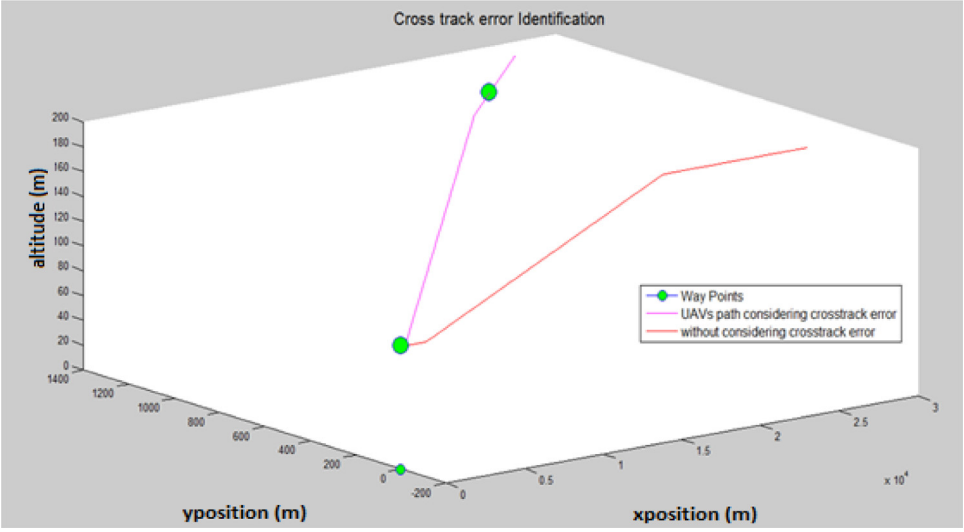


Fig. 6. Avoiding Cross track error.

- Determine the position of obstacle with respect to way-points
- Determine collision possibility
- Update waypoint list including new waypoints identified to avoid collision

The pseudo code for main module is shown in Fig. 5.

3.4. Flight controller module

The Flight Controller Module is responsible for navigating UAV along the predetermined trajectory. The Way Points *WP* along the intended trajectory for UAV is obtained from the Main module to issue control commands to the UAV to navigate it along the original path. The cross track error feedback identifies the deviation in trajectory from the defined path and based upon the error, the UAV is navigated by this module towards the intended trajectory.

When the collision avoidance module determines alternate waypoints to avoid collision, the flight controller in turn calculates the necessary changes in flight heading to navigate through the new waypoint list.

3.5. Cross track error avoidance module

The cross track error is a measure of deviation in UAV's desired path and is converted to an adjustment of the bearing. Once bearing is known, then how much the UAV has to roll to reach that bearing is decided by Bank angle [41]. Fig. 6 shows cross track error correction.

$$\tan\varphi = \frac{v^2}{gR} \tag{13}$$

$$\text{Rate of Change of Heading} = g\tan\varphi/v \tag{14}$$

where, $g = 9.81\ \text{m/s}^2$-acceleration due to gravity; φ-Banking angle

v—velocity of UAV; R-Radius of turn

$$\text{Cross track error } DD = \frac{(A * xpos + B * ypos + C)}{\sqrt{AA^2 + BB^2}} \tag{15}$$

where, $A = Y_{fin} - Y_{st}$, $B = X_{st} - X_{fin}$, $C = X_{st} * (Y_{st} - Y_{fin}) + Y_{st} * (X_{fin} - X_{st})$, X_{st}, X_{fin} -starting, final waypoints in x coordinates, Y_{st}, Y_{fin} -starting, final waypoints in y coordinates, x_{pos}, y_{pos} -UAV's position in x and y coordinates.

3.6. Leg change over module

The starting waypoints of UAV are denoted as (X_{st}, Y_{st}) and the final waypoints is denoted as (X_{fin}, Y_{fin}). The flight controller finds the distance between the current position of UAV and starting waypoints and calculates the distance between starting and final waypoints.

$$UAV\ Distance = \sqrt{(x_{pos} - X_{st})^2 + (y_{pos} - Y_{st})^2 + (z_{pos} - alt)^2} \tag{16}$$

$$Waypoint\ Distance = \sqrt{(X_{fin} - X_{st})^2 + (Y_{fin} - Y_{st})^2 + (z_{fin} - alt)^2} \tag{17}$$

where, z_{pos}, Z_{fin}, alt — current position, final waypoints in z coordinates and altitude

When the current leg of travel is over, the trajectory has to be shifted to the next leg. The UAV changes leg when the difference between *Waypoint Distance* and *UAV Distance* is less than 1 km. The flight controller module guides the UAV to turn its head currently pointing to (X_{fin}, Y_{fin}) towards next waypoint. This is mainly done to avoid sharp changes in the trajectory of UAV. The aileron is responsible for how much the aircraft has to roll to turn the head of an aircraft. Aileron command is computed as an error between roll command and current roll.

$$Aileron\ command = -(rollcommand - \varphi) * 20 \tag{18}$$

$$roll\ command = roll\ command + ((DD \times 0.001 + DD_{rate} \times 0.05) \times 0.1) - roll\ command \times 0.01 \tag{19}$$

When the UAV is at ground level, the roll command and the previous cross track error value (DD_{old}) are assumed to be zero. The current DD is considered as DD_{old} for next iteration.

$$DD_{rate} = \frac{(DD - DD_{old})}{0.01} \tag{20}$$

1. *procedure stationary_moving*
2. Find UAV position and range (distance between UAV and obstacle)
3. Calculate difference between two positions (change_UAV_ position) and two ranges (change_in_ range) obtained for two instants of time
4. **if** change_in_range > (change_UAV_ position)
5. **then** *moving* **else** *stationary*
6. **end if**
7. end *procedure*

Fig. 7. Algorithm 2: *stationary_moving*.

1. *procedure continuous_obstacle*
2. **if** distance between detected obstacles < 1 Km
3. **then** obstacles are continuous
4. **else** obstacles are single and scattered
5. **end if**
6. end *procedure*

Fig. 8. Algorithm 3: *continuous_obstacle*.

1. *procedure near_end_waypoint*
2. Calculate Distance (distance between X_{fin} and obstacle)
3. **if** Distance < safety distance (1 Km)
4. **then** obstacle is near end waypoint
5. **end if**
6. end *procedure*

Fig. 9. Algorithm 4: *near_ end _waypoint*.

1. *procedure* ***obs_bet_wps***
2. Calculate Distance1(distance between X_{fin} and UAV position)
3. Calculate Distance2(distance between current obstacle and UAV position)
4. **if** Distance1 > Distance2
5. **then** obstacle is between waypoints
6. **end if**
7. end *procedure*

Fig. 10. Algorithm 5: *obs_bet_wps*.

3.7. Identification of type and position of obstacle

The obstacles within radar range are to be categorized as stationary or moving as given in Fig. 7 to implement appropriate avoidance strategy. If the distance between UAV and obstacle reduces rapidly it is categorized as moving obstacle. Further, the obstacles identified as stationary need to be categorized as single or continuous as given in the algorithm ***stationary_ moving*** of Fig. 8. The obstacles close to each other (within 1 km) are clustered together as continuous or voluminous obstacles.

The position of obstacle is to be determined for taking appropriate avoidance action. The algorithm ***near_ end _waypoint*** in Fig. 9 checks whether the obstacle is nearby the waypoint to be reached. If so, the waypoint could not be reached and is replaced by an alternate waypoint.

The procedure ***obs_bet_wps*** in Fig. 10 checks whether the identified obstacle is positioned between starting and end waypoints while travelling along the leg. New waypoints are to be identified to avoid collision and the trajectory to be changed via the new waypoint to reach the destination.

3.8. Collision detection module

Collision exists if collision distance $CD < RS$, accounting for errors in radar measurements where RS is the Radius of Safety Sphere around obstacle

$$CD = \sqrt{(x_{pos} - x_o)^2 + (y_{pos} - y_o)^2 + (z_{pos} - z_o)^2} \tag{21}$$

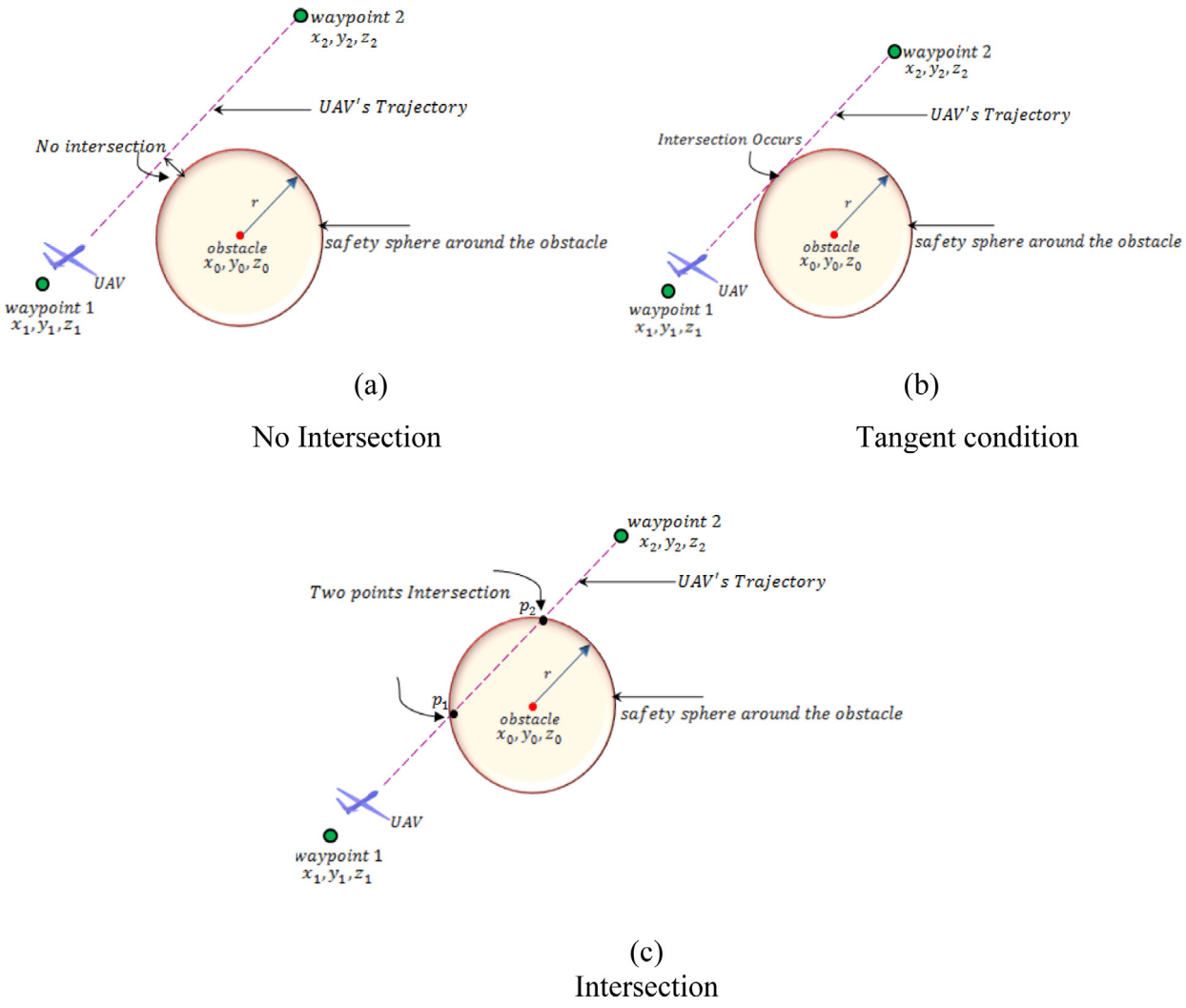


Fig. 11. Trajectory and Safety Sphere Intersection.

where ($x_{pos}, y_{pos}, z_{pos}$) represents the current position of UAV and (x_o, y_o, z_o) represents the position of obstacle.

The proposed algorithm identifies collision before the condition $CD < RS$ is reached. Once the position of obstacles is received from radar, the obstacles along the UAV's trajectory are identified by using line and sphere intersection equations (26). The presence of obstacles along the UAV's trajectory confirms future collision and an alternate trajectory is identified immediately including new way point. In this way, collision is avoided much earlier thus making it suitable for real time applications.

The parametric line equations which represent UAV trajectory are

$$x = x_1 + (x_2 - x_1)\, t,$$

$$y = y_1 + (y_2 - y_1)\, t, \tag{22}$$

$$z = z_1 + (z_2 - z_1)\, t.$$

The sphere equation for representing the safety sphere of obstacle is given as

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \tag{23}$$

Substituting the line equations (22) in sphere equation (23) gives a quadratic equation of the form $at^2 + bt + c = 0$. The intersection points p_1, p_2 between and sphere and line is determined by substituting t in parametric line equations. The solution for t is

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{24}$$

where, $a = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$

$$b = 2\left[(x_2 - x_1)(x_1 - x_0) + (y_2 - y_1)(y_1 - y_0) + (z_2 - z_1)(z_1 - z_0)\right]$$

$$c = x_0^2 + y_0^2 + z_0^2 + x_1^2 + y_1^2 + z_1^2 - 2(x_0x_1 + y_0y_1 + z_0z_1) - r^2 \tag{25}$$

r represents the radius of safety sphere RS around the obstacle.

Collision of UAV with obstacle is identified by the following conditions as shown in Fig. 11.

Condition for intersection (collision): $b^2 - 4ac > 0$

Condition for tangency (collision): $b^2 - 4ac = 0$

No intersection (No Collision): $b^2 - 4ac < 0$ (26)

The ***collision_possibility*** module identifies the possibility of collision. The steps are shown in Fig. 12.

For moving obstacle, collision possibility is reaffirmed within a safe period of time since it might change its route anytime. The procedure *Confirm_Collision* is shown in Fig. 13.

1. *procedure collision_possibility*
2. Initialize radius of safety sphere *RS* around obstacle
3. **if** UAV's trajectory intersects safety *sphere*
4. **then** collision is **true**
5. **else** collision is **false**
6. **end if**
7. end *procedure*

Fig. 12. Algorithm 6: *collision_possibility*.

4. Collision avoidance module using PSO

The Avoidance Module establishes communication with flight controller and radar unit to get details of UAV and obstacles. Based on the current flight's position, its trajectory and the position of obstacles, the avoidance module identifies new optimized waypoints which is passed on to the Flight Controller Module to navigate the UAV safely in alternate path.

Particle Swarm Optimization (PSO) is an evolutionary computation algorithm which was developed by Kennedy and Eberhart (1995), inspired by birds flocking and fish schooling. The PSO maintains the population of particles (solutions) with randomized velocity (V^k) and position (X^k) [12]. The initial objective function (f) is calculated using initial solutions. The initial solution called seed points are obtained along the circumference of safety sphere around obstacle using tangential equation (27).

$$t_1 = \pm\cos^{-1}\left(\frac{-rx_0 \pm y_0\sqrt{x_0^2+y_0^2-r^2}}{x_0^2+y_0^2}\right) \tag{27}$$

where (x_0, y_0) is the position of obstacle and r is the radius of safety sphere around it.

The algorithm ***Find_InitialNew_Waypoints*** for finding initial seed points (10 particles) for PSO is given in Fig. 14.

Each particle keeps its best solution as *pbest* and for every iteration the best solution is compared with the global search space solution *gbest*. If *pbest* is better than *gbest*, then *pbest* is assigned as*gbest*. The final solution is *gbest*. At each iteration, velocity is updated within bounded limits thereby updating position [12] and the objective function is calculated with new velocities. The velocity and position are updated as in equations below,

$$V^{(k+1)} = wV^k + C_1 rand_1(p_{best}{}^k - X^k) + C_2 rand_2(g_{best}{}^k - X^k) \tag{28}$$

$$X^{(k+1)} = X^k + V^{(k+1)} \tag{29}$$

$$where\ w = w_{max} - \frac{(w_{max} - w_{min}) \times k}{I} \tag{30}$$

where, I — Total iterations; k — current iteration number, V^k — current velocity,

X^k — current position, C_1, C_2 — scaling factors, $rand_1$, $rand_2$ — random values $pbest^k$ — personal (iteration) best value, $gbest^k$ — global best value.

The algorithm *searchzone_limit* to keep velocity within bounded limits is shown in Fig. 15. This step limits deviation in search space from current position of UAV while finding new solution.

1. *procedure Find_InitialNew_Waypoints*
2. Find random solutions around safety sphere using tangent equations (27)
3. Return set of 10 initial waypoints as particles for PSO algorithm
4. **end** *procedure*

Fig. 14. Algorithm 8: *Find_InitialNew_Waypoints*.

1. *procedure searchzone_limit*
2. Initialize $saferegionLimit$
3. do for all particles
4. $maxbound = X^K + saferegionLimit$
5. $minbound = X^K - saferegionLimit$
6. **end** *procedure*

Fig. 15. Algorithm 9: *searchzone_limit*.

Table 2
PSO parameters.

Parameters	Value
Total number of particles	N=10
Number of parameters	$2 = [x_p, y_p]$
Number of iterations	I=30
Stopping Criteria	Iterations or change in fitness<200 for continuous 5 iterations
Weighing factor	$w_{max} = 0.09,\ w_{min} = 0.02$
Scaling factor	$C_1 = C_2 = 2$
rand	Random values between (0, 1)
Initial velocity	$V^k = 0$ for k=1
Limiting factor	alpha=0.1
saferegionLimit	4 km

The objective function represented in Eq. (6) is minimized by PSO algorithm subject to the constraints in (12). The parameter setting for PSO algorithm is listed in Table 2. The range of PSO parameters were obtained from literature [22,23,44,45] and the final parameters were chosen after several trials and runs within the range that suits real time obstacle avoidance scene.

The algorithm ***PSO_Optimized_Waypoint*** in Fig. 16 details the procedure of obtaining optimized waypoint with maximum safety and minimum path length.

5. Results and discussion

5.1. Experimental setup

The proposed PSO-CA obstacle avoidance system is simulated using MATLAB 2014b software in Windows 7 Personal Computer, i3 core, 2.27 GHz with 4 GB RAM. As there are no widely accepted benchmark problems for UAV path planning [3], this work designed several scenarios with random stationary obstacles ranging from 1 to 100 and hill like obstacles along the path of UAV. Further 6-DOF UAVs moving at 15 m/s are also introduced as moving obstacles to check the efficiency of the proposed algorithm.

Experimental procedure is as follows:

1. *procedure Confirm_Collision*
2. Calculate Collision Distance CD=Distance3 (distance between obstacle and UAV)
3. Calculate Distance3new ((distance between moving obstacle and UAV position after some interval of time)
4. **if** Distance3new<Distance3 **then**
5. Collision confirmed
6. **end if**
7. end *procedure*

Fig. 13. Algorithm 7: *Confirm_Collision*.

1. *procedure PSO_Optimized_Waypoint*
2. Initialize PSO parameters, 10 particles (seed waypoints x_p,y_p obtained from *Find_InitialNew_Waypoints*), velocity of 10 particles, *maxiter* (maximum number of iterations)
3. **for** each particle do
4. Evaluate fitness value $f(x_p,y_p)$ based on Objective Function or Fitness function

$$f = W_t f_1 + W_0 f_2$$

where $f_1 = ((x_p - x_{pos})^2 + (y_p - y_{pos})^2)^{1/2} + ((x_{fin} - x_p)^2 + (y_{fin} - y_p)^2)^{1/2}$

$$f_2 = \left(\sum_{k=1}^{N_0}(MXRSK^k - dis^k)\right)^2$$

5. **end**
6. Find minimum fitness value f among the 10 particles and set its position (x_p,y_p) as *pbest*
7. *gbest* = p*best.*
8. Start iteration *iter=1*
9. **Repeat** for each iteration
10. **for** each particle do

$$w = w_{max} - \frac{(w_{max}-w_{min})\times current\ iteration}{Total\ Iteration}$$

11. Update velocity of each particle

$$V^{(k+1)} = wV^k + C_1 rand_1({p_{best}}^k - X^k) + C_2 rand_2({g_{best}}^k - X^k)$$

12. Obtain *maxbound* and *minbound from searchzone_limit*
13. $velmax = alpha * (maxbound - minbound)$
14. $velmin = -velmax$
15. Update velocity within bounds

$V^{(k+1)} = max\ (V^{(k+1)}, velmin)$ if $V^{(k+1)} < velmin$

$V^{(k+1)} = min(V^{(k+1)}, velmax)$ if $V^{(k+1)} > velmax$

16. **Update position** (x_p,y_p) of each particle: $X^{(k+1)} = X^k + V^{(k+1)}$
17. Evaluate fitness value *f*
18. **end**
19. Find minimum fitness value $fmin$ for all 10 particles with new position
20. **if** *f min< f(pbest)* **then**
21. *pbest*=Current position (x_p,y_p)
22. **end if**
23. **if** *f(pbest)<f(gbest)*
24. *gbest=pbest*
25. **end if**
26. ***iter=iter+1***
27. **Until** ***iter*** **=** ***maxiter*** or stopping criteria reached
28. *Optimized solution=gbest*
29. **end** *procedure*

Fig. 16. Algorithm 10: *PSO_Optimized_Waypoint*.

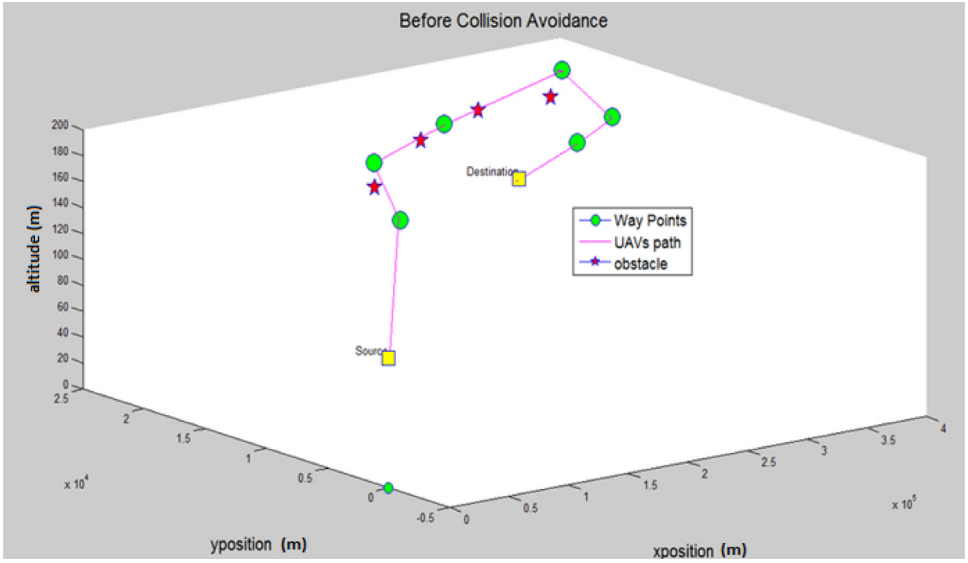


Fig. 17. UAV's trajectory without collision avoidance.

1. 6-DOF UAV was designed in Matlab and navigated along designated waypoints. The set of 8 waypoints (position in metres) for the experiments conducted is shown below

$x = [0\ 20000\ 50000\ 100000\ 200000\ 350000\ 300000\ 270000]$

$y = [0\ 1000\ 6500\ 11000\ 15000\ 20000\ 11000\ 10500]$

$z = [100\ 200\ 200\ 200\ 200\ 200\ 200\ 200]$

2. The trajectory between first two waypoints is considered as leg 1 and the UAV travels in straight line avoiding cross

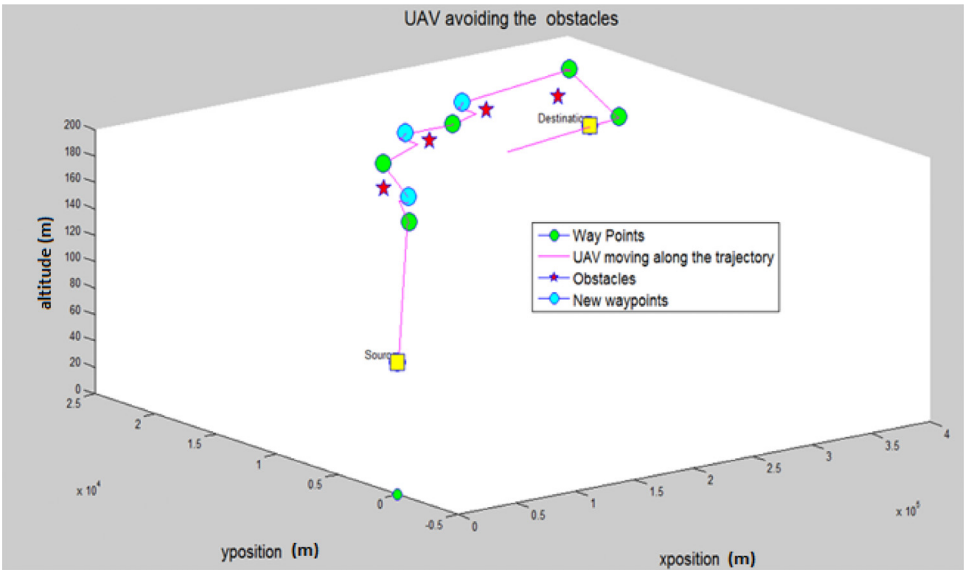


Fig. 18. UAV's trajectory with collision avoidance.

track error from waypoint 1 to waypoint 2. Radius of Safety Sphere around obstacle is initialized as 1 km

Waypoint 1 = [0, 0, 100]
Waypoint 2 = [20000, 1000, 200]

3. Moving obstacles are designed using 6-DOF equations in separate functions and navigated simultaneously by passing their waypoints from main program. Radar module obtains the position of moving obstacle each and every second from the function executed in parallel.
4. All the stationary obstacles are generated (Refer Appendix) and the radar module tracks each and every obstacle once it falls within its range
5. All the moving and stationary obstacles within radar range are checked for the feasibility of collision during travel along leg 1.
6. The obstacles with collision possibility are immediately avoided by updating waypoints list including alternate waypoints.
7. The leg is increased and the UAV travels along next leg from second waypoint to third waypoint in the waypoint list
8. The process continues until all the legs are traversed and the UAV reaches the final waypoint

Three types of experiment are carried out namely

1. **UAV avoiding stationary definite and random obstacles**
   This experiment is conducted to show that the UAV is capable of avoiding discrete single obstacles that pops up along its path
2. **UAV avoiding stationary voluminous obstacles like hill**
   The purpose of this experiment is to show that the UAV is capable of identifying voluminous obstacles like hill based on radar data and avoid it
3. **UAV avoiding moving obstacles**
   This experiment shows that any moving threat in UAV's path could be avoided

The position of obstacles and waypoints for the results shown in Figures are given in Appendix.

5.2. UAV avoiding stationary obstacles

Fig. 17 shows the path of UAV without the collision avoidance scheme which depicts the occurrence of collision. Fig. 18 shows the safe navigation of UAV with collision avoidance.

5.3. UAV avoiding voluminous obstacles like hill

All the obstacles with distance less than 1 km between them are grouped together and treated as continuous or voluminous obstacles. The boundary/terminal obstacles of the group on both sides are identified to find the alternate path. Only those nearby obstacles that come under radar view simultaneously are grouped together. In such a case, only part of the hill viewed by radar is considered for avoidance while the remaining part is considered after they fall in subsequent radar view. Fig. 19 shows the UAV avoiding continuous obstacles.

The trade-off between time and optimized path of the algorithm with increase in number of obstacles is studied for the path with waypoints shown in Section 5.1 and the results are tabulated in Table 3. The time taken by the UAV to fly over the designated path along with the time taken by the algorithm is reported in Table 3. The time and the optimized path length are represented as the average of successful runs out of 10 runs.

The collision detection and identification of alternate waypoint is done immediately once the obstacle is visible to the radar. As the maximum time taken for algorithm to find out alternate waypoint after detecting collision and updating the waypoint list is around 28 s for a single obstacle, the UAV is very safe as it is flying at a speed of 15 m/s. The time spent by PSO does not affect the maximal speed of UAV as alternate waypoints are identified earlier but limits the density of obstacles. For multiple obstacles with collision threat, the proposed algorithm avoids the first obstacle seen by radar and subsequently the remaining ones after they falls within radar range. For simultaneous view of many obstacles, the closest one is avoided first. An alternate waypoint identified to avoid one obstacle may be near another one obstacle which may come under radar view later. In such a case that alternate waypoint identified is skipped by choosing another safe waypoint considering this new obstacle.

The algorithm might not have identified alternate waypoints for all the obstacles since some of them might have been away

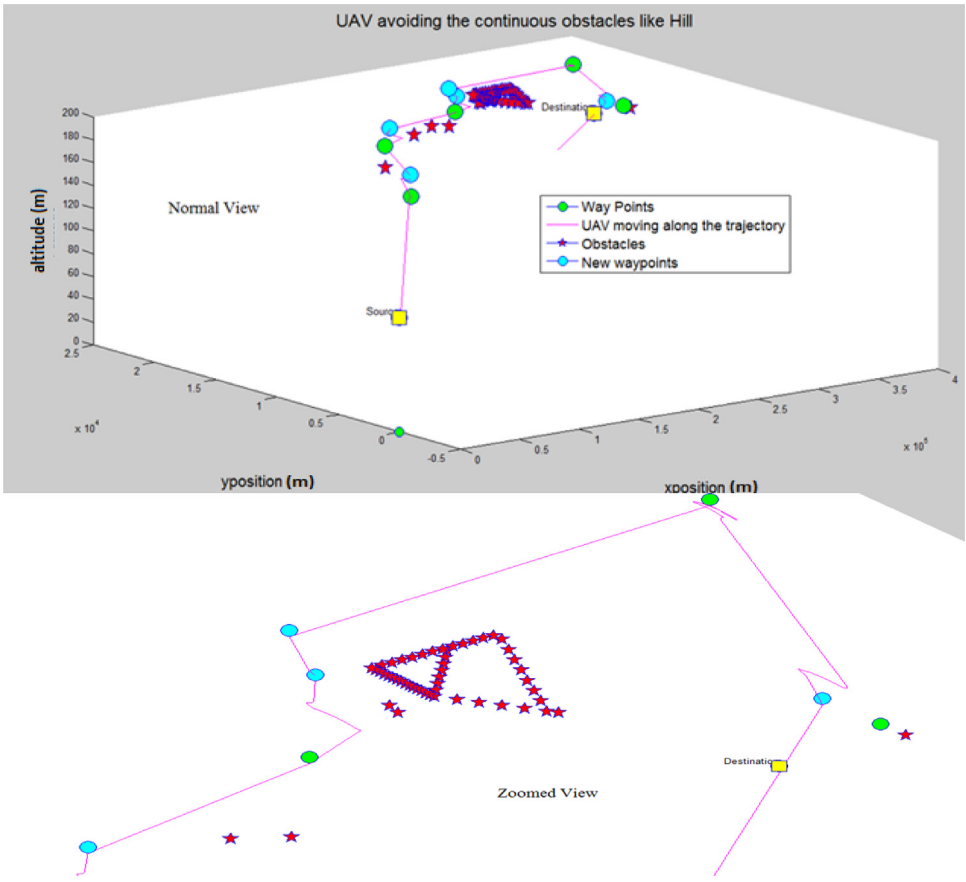


Fig. 19. UAV avoiding hill like obstacles.

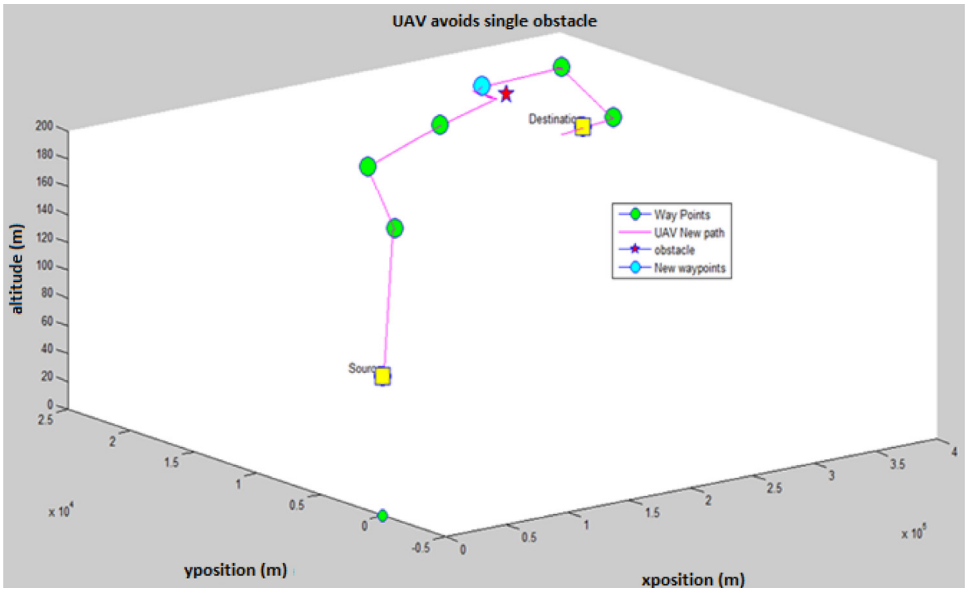


Fig. 20. UAV avoiding single obstacle via shortest path.

from the UAV trajectory and some might have been avoided automatically while finding alternate path for the obstacles faced first. As the obstacle density increases, it becomes tough to find alternate waypoints, as the alternate waypoint identified may be near other obstacles and hence the algorithm results in many new waypoints to find safe path. This could be witnessed from the sudden increase in time for 70 obstacles compared to 60 obstacles. As the number of obstacles increases above 70, few runs could not identify safe path due to overcrowding of obstacles.

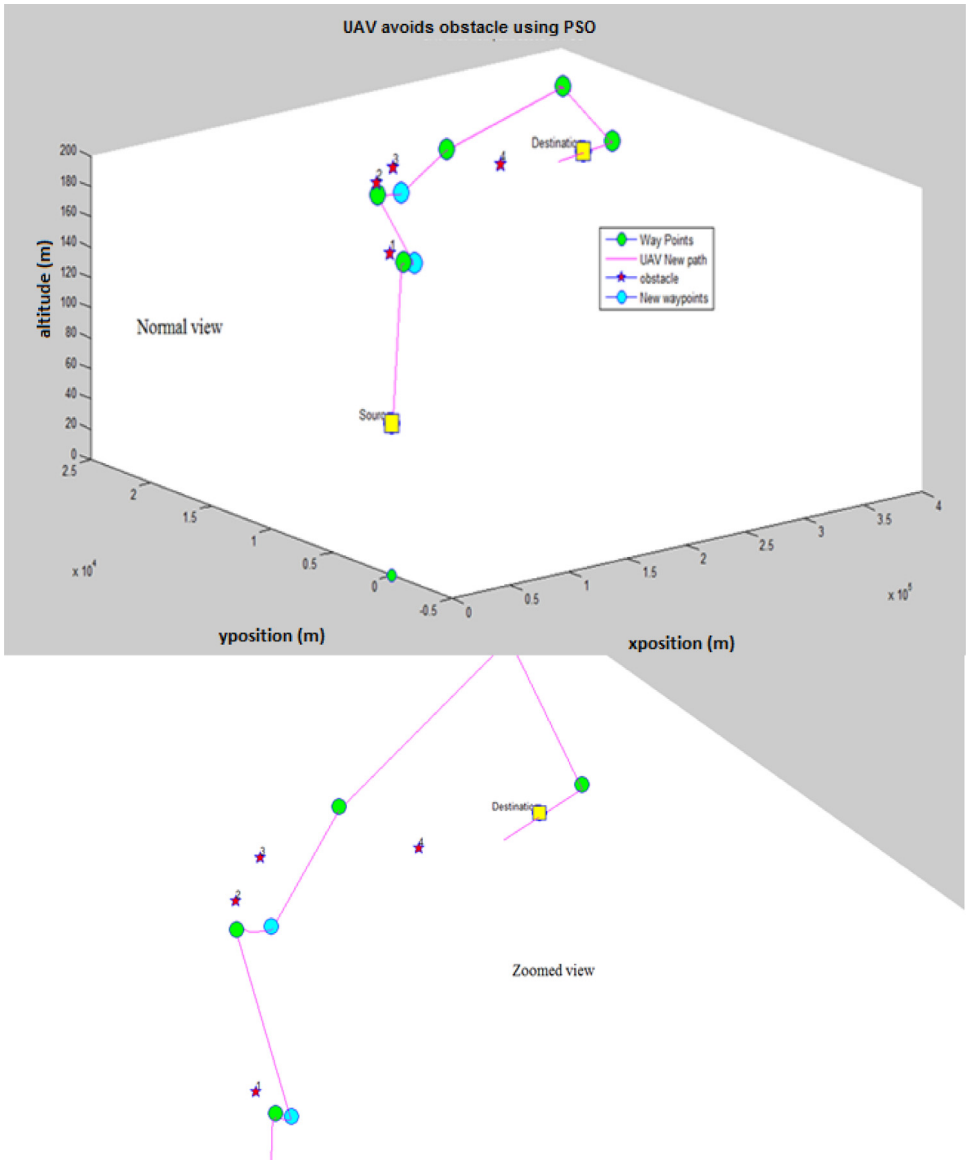


Fig. 21. No obstacle along shortest path.

5.4. Significance of PSO in collision avoidance

This section justifies the significance of PSO optimization algorithm in determining the new way point for alternate trajectory while avoiding obstacle. Several cases are discussed where the PSO optimization decides the safer path of travel in an optimized way. The algorithm focuses on shorter path of travel while ensuring maximum safety.

Case (i)

If there is only one obstacle between 2 waypoints as in Fig. 20, then optimization algorithm decides which way to choose either left or right. As there is no other danger on both paths, the optimization is left with the decision to choose the shortest path among them. Once the new way point is identified to avoid obstacle, the original trajectory is modified dynamically and the updated waypoint list includes the new way point.

Initial trajectory waypoint list WP= $\{WP_1, WP_2, WP_3, WP_4, WP_5, WP_6, WP_7, WP8\}$

The initial trajectory which contains 8 way points is modified to include the new waypoint ***WP12new***, when an obstacle lies in the trajectory of UAV between WP_1 and WP_2.

Modified trajectory waypoint list $WPnew$= $\{WP_1$, ***WP12new,*** $WP_2, WP_3, WP_4, WP_5, WP_6, WP_7, WP8\}$

Case (ii)

When there is more than one obstacle between two waypoints, the algorithm has to choose the shortest path while avoiding the first obstacle encountered, but at the same time it should not hit the other obstacles. Hence, the algorithm is designed in such a way that if other obstacle comes along the shortest path, then the next route is considered even though it is lengthy. Fig. 21 shows the UAV turning right to avoid obstacle numbered 2 along the shortest path. When an obstacle numbered 3 (within radar range) comes along the shortest path as in Fig. 22, UAV turns left avoiding shortest path which is not safe. As the obstacle numbered 4 comes under radar view after avoiding obstacle 2, a new way point is chosen to avoid obstacle 4.

Case (iii)

If an obstacle lies close to the waypoint, it is not safe to reach that waypoint. Hence, an alternate waypoint is identified by PSO instead of that unsafe waypoint. Fig. 23 shows the UAV skipping such unsafe waypoint and navigating along the alternate path.

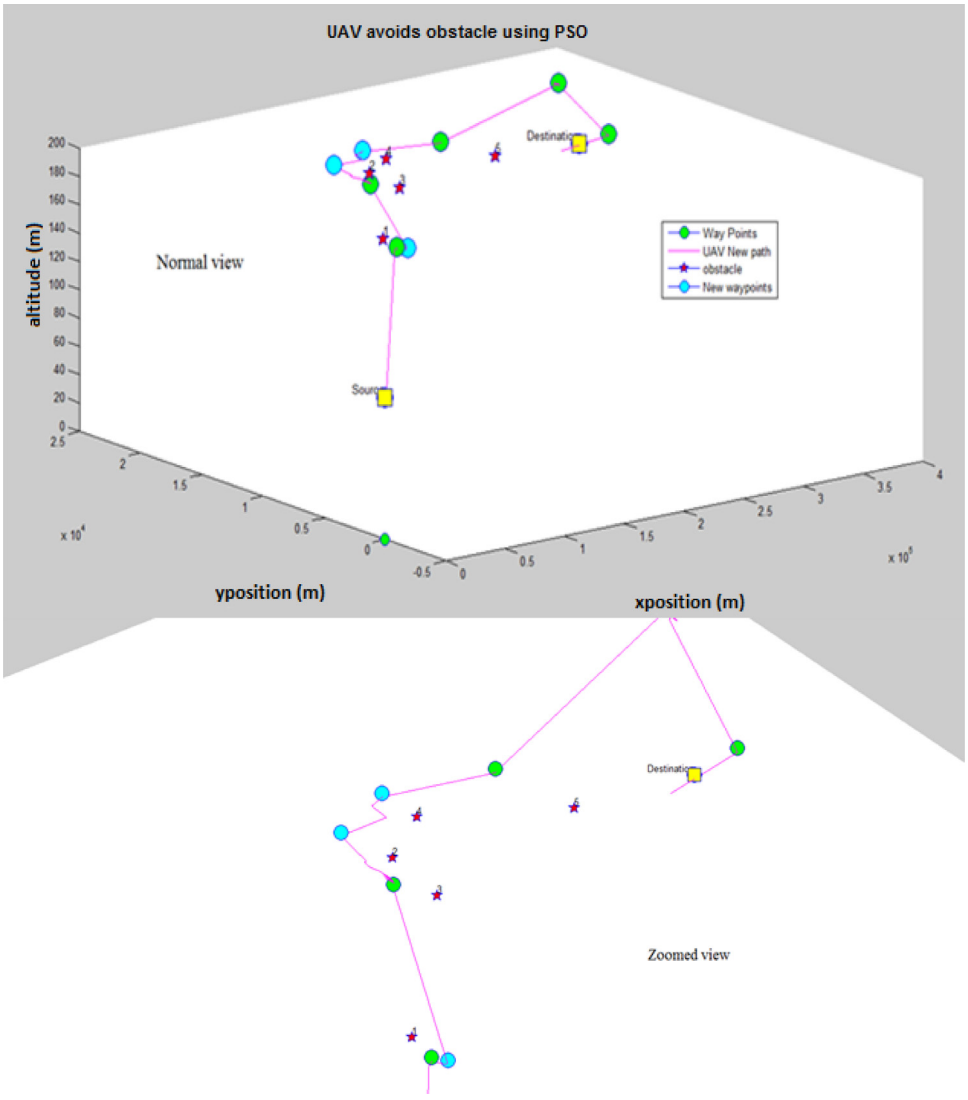


Fig. 22. Obstacle along shortest path.

Table 3
Performance metric.

No of obstacles	UAV flying time in minutes	PSO-CA algorithm running time in seconds	Optimized path length in metres
No any obstacle	480.896933	–	431630
1	481.085145	28.5469	431650
10	482.588177	29.7656	432154
20	493.687412	35.4219	442063
30	492.450745	43.2813	442948
40	493.763296	62.0313	443846
50	494.283566	63.5313	444640
60	495.325971	69.4876	445548
70	498.768901	93.6453	448653
80	501.555750	95.9354	450900
90	504.345905	98.5632	453734
100	509.675489	117.3524	458532
40 discrete &1 Hill obstacle	500.786454	65.3874	447231
40 discrete & 2 moving obstacles	496.232877	66.7465	444922

5.5. Convergence of PSO

The stopping criteria depends on both change in fitness value and number of iterations. If there is no change in fitness value continuously for more than 5 iterations, the algorithm is terminated. Fig. 24 shows the fitness value of 10 particles (population size=10) in a single iteration for a sample run and its best fitness value Pbest.

The fitness function started decreasing but failed to converge for 10 and 20 iterations as in Fig. 25(a) and (b). Fig. 25(c) shows the convergence for 30 iterations for a sample run. After so many trial runs, it is observed that the algorithm converges within a maximum of 30 iterations for many runs. Hence, the total number

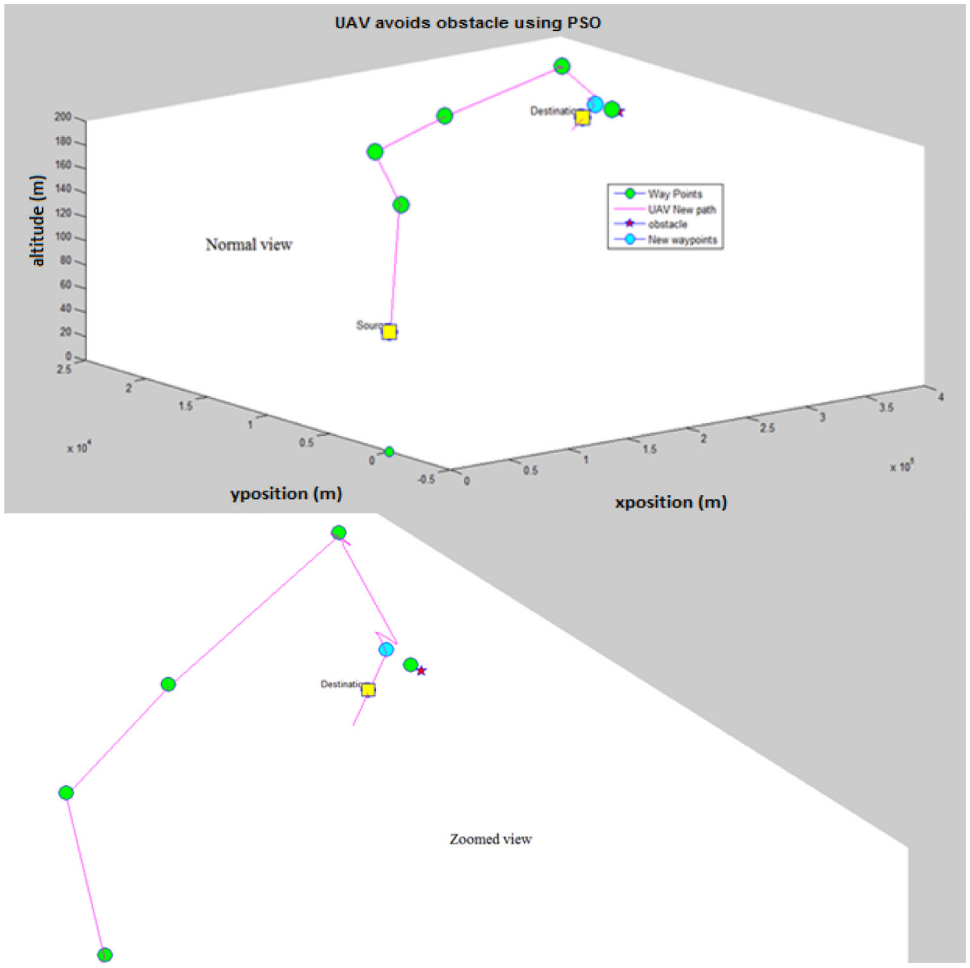


Fig. 23. UAV skips unsafe waypoint.

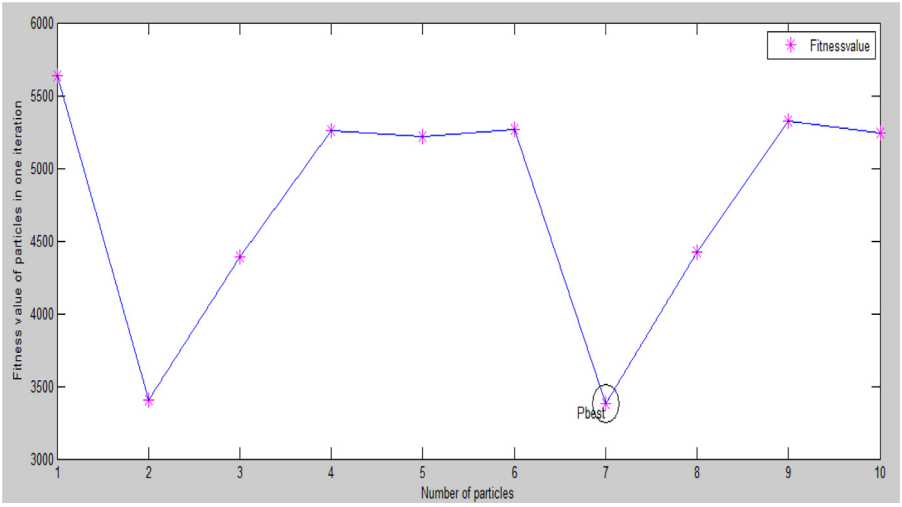
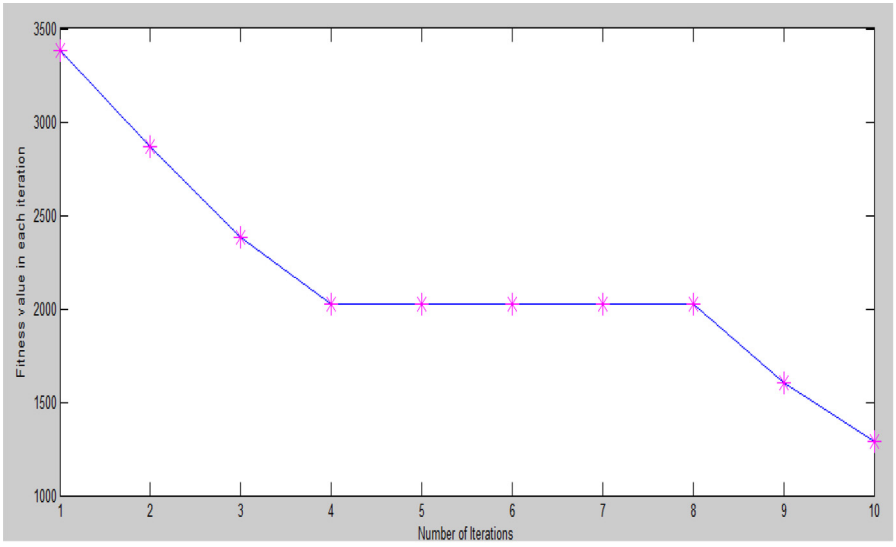


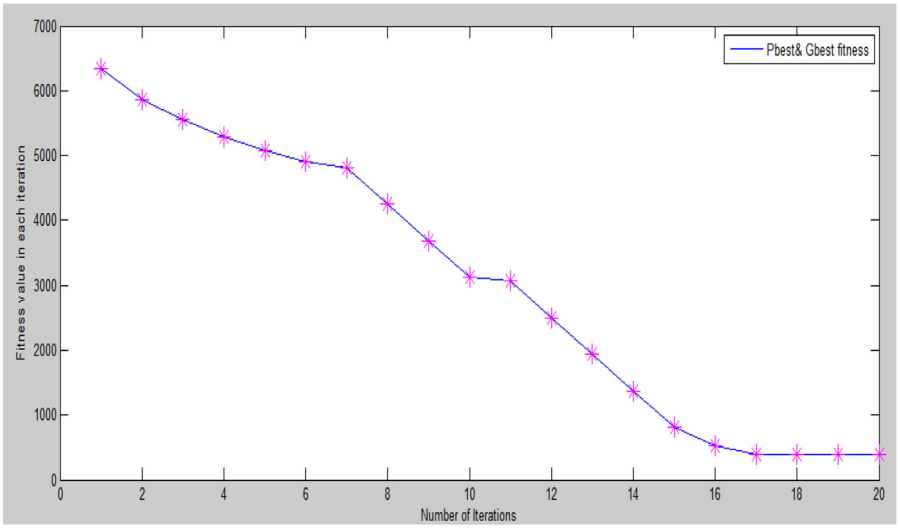
Fig. 24. Fitness value for single iteration (Population size=10).

of iterations is set as 30. In order to reduce the time of run to suit real time, the algorithm is terminated even before the prescribed iterations if the change in fitness value for continuous 5 iterations is less than a prescribed limit (<200).

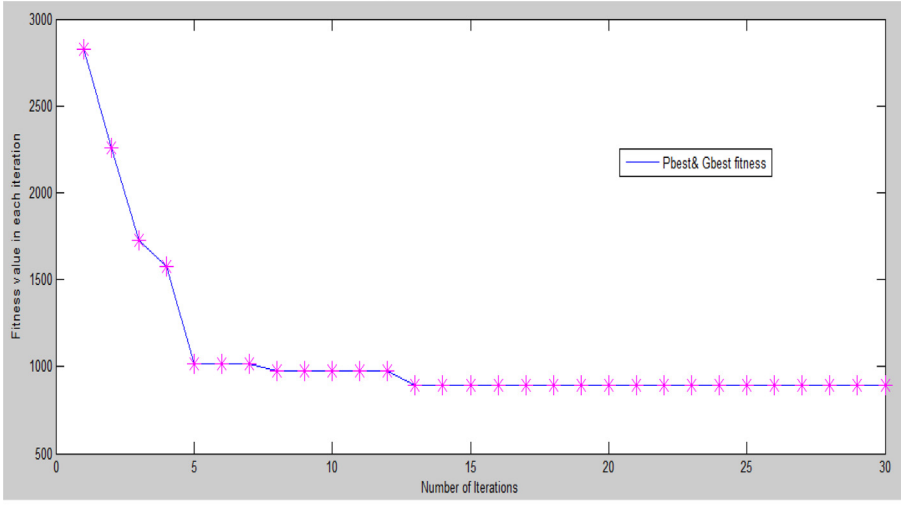
For convergence proof it is enough to show that a random search algorithm will at least converge to a local minimum if the algorithm satisfies the algorithm condition and the convergence condition [46]. The definitions for convergence can be referred from [46]. The algorithm condition specifies that the new solution suggested by the algorithm is to be no worse than the current solution. From Fig. 25, it is understood that the new solution is better than current solution and hence algorithm condition is satisfied.



(a)Iterations=10



(b) Iterations =20



(c) Iterations =30

Fig. 25. Convergence of algorithm.

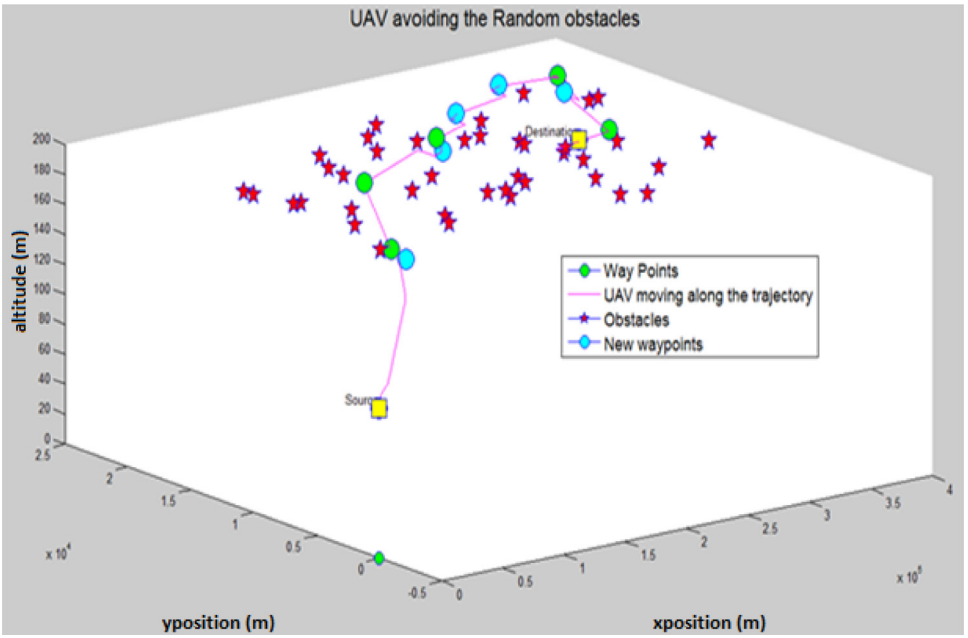


Fig. 26. Collision Avoidance for random obstacles.

Convergence condition *For any* $x_t \in S$ *there exists a* $\gamma > 0$ *and an* $0 < \eta \leq 1$ *such that*

$$\mu_t\left(dist\left(x_{t+1}, R_\in\right) \leq dist\left(x_t, R_\in\right) - \gamma \text{ or } x_t \in R_\in\right) \geq \eta \tag{31}$$

An algorithm is a local optimization algorithm if a non-zero η can be found such that at every step the algorithm can move x closer to the optimality region by at least distance γ, or x is already in the optimality region with a probability greater than or equal to η. Fig. 25 shows that at each step the algorithm moves closer to the optimality region by at least distance γ (100), or x is already in the optimality region.

5.6. Algorithmic limits

Random obstacles are generated using Monte Carlo simulation and the proposed system steered UAV safely for 10 runs successfully until it faced a maximum of 70 stationary obstacles along the path. A normal (bell curve) probability distribution with mean or expected value as waypoints and standard deviation 3000 is specified for simulation of obstacles. Fig. 26 depicts the performance for 40 obstacles. The algorithm performs well for all runs up to 70 obstacles.

5.7. UAV avoiding moving obstacles

The radar unit of UAV identifies the position of moving obstacles once they enter into its range. The sudden reduction in the range of obstacle identifies it is a moving one and when the distance between the UAV and the intruder falls below a safety range (8 km in this work), the UAV changes its trajectory to avert any collision. It either turns left or right sensing the trajectory of moving obstacle. If the obstacle is heading towards UAV as in Fig. 27, the UAV turns either left or right. If the obstacle is heading towards left, the UAV turns right as in Fig. 28 and if the intruder is moving towards right, our UAV turns left. If the UAV's trajectory intersects the safety sphere around the moving obstacle collision is detected and the alternate waypoint is identified similar to single stationary obstacle treating the current position of incoming UAV as the position of stationary obstacle. The alternate waypoint beyond a safety sphere of 2 km around the moving obstacle is identified and the UAV is directed towards that new waypoint. As the time taken to identify alternate waypoint by PSO algorithm is around 28 s, by that time both the UAV and moving obstacle might have travelled around 500 m. This does not affect the safe journey of UAV in real time. A sample video of an UAV travelling along waypoints avoiding both stationary and moving obstacle is available at https://youtu.be/tKfcbSE9VVA. The movement of UAVs with their change in position is represented using a graphically designed flight like object. This moving obstacle avoidance algorithm is tested only for other UAVs and not for fast moving aircraft.

6. Comparison with other planners

In this paper, the proposed PSO-CA algorithm was compared with two heuristic techniques namely a population based Genetic Algorithm (GA) and a solo search probabilistic based Simulated Annealing (SA) algorithm. The parameters of GA are Generation — 30, crossover rate — 0.75, mutation rate — 0.008, selection — Roulette Wheel, Crossover-single point and that of SA are iterations 30, Initial Temperature — 100, Acceptance Function — Simulannealbnd. The stopping criteria (iterations or change in fitness value less) is same for all the three algorithms.

Some of the state of the art research have addressed problems similar to this paper, but it is very difficult to compare such works as the proposed obstacle avoidance algorithm avoids obstacles between two waypoints (along one leg of travel) and has not considered all obstacles together. Also, the dependency of the algorithm on radar data makes the present work computationally different from other existing works. But the optimization part of the algorithm could be compared with other latest optimization algorithms suggested in recent literature. The optimization algorithm suggested by Xiangyin et al. in [21] namely the Phase angle encoded Mutation adopted Fruit Fly Optimization algorithm θ-MAFOA is also included in the comparison result. The iterations and population size are set the same as our PSO-CA algorithm 30 and 10 respectively. Literature [21] has reported that mutation rate less than 1 works well (between 0.5 and 0.9) and hence a value of 0.8 is chosen. The authors in [21] have added phase angle information along with location in x, y coordinates for improving

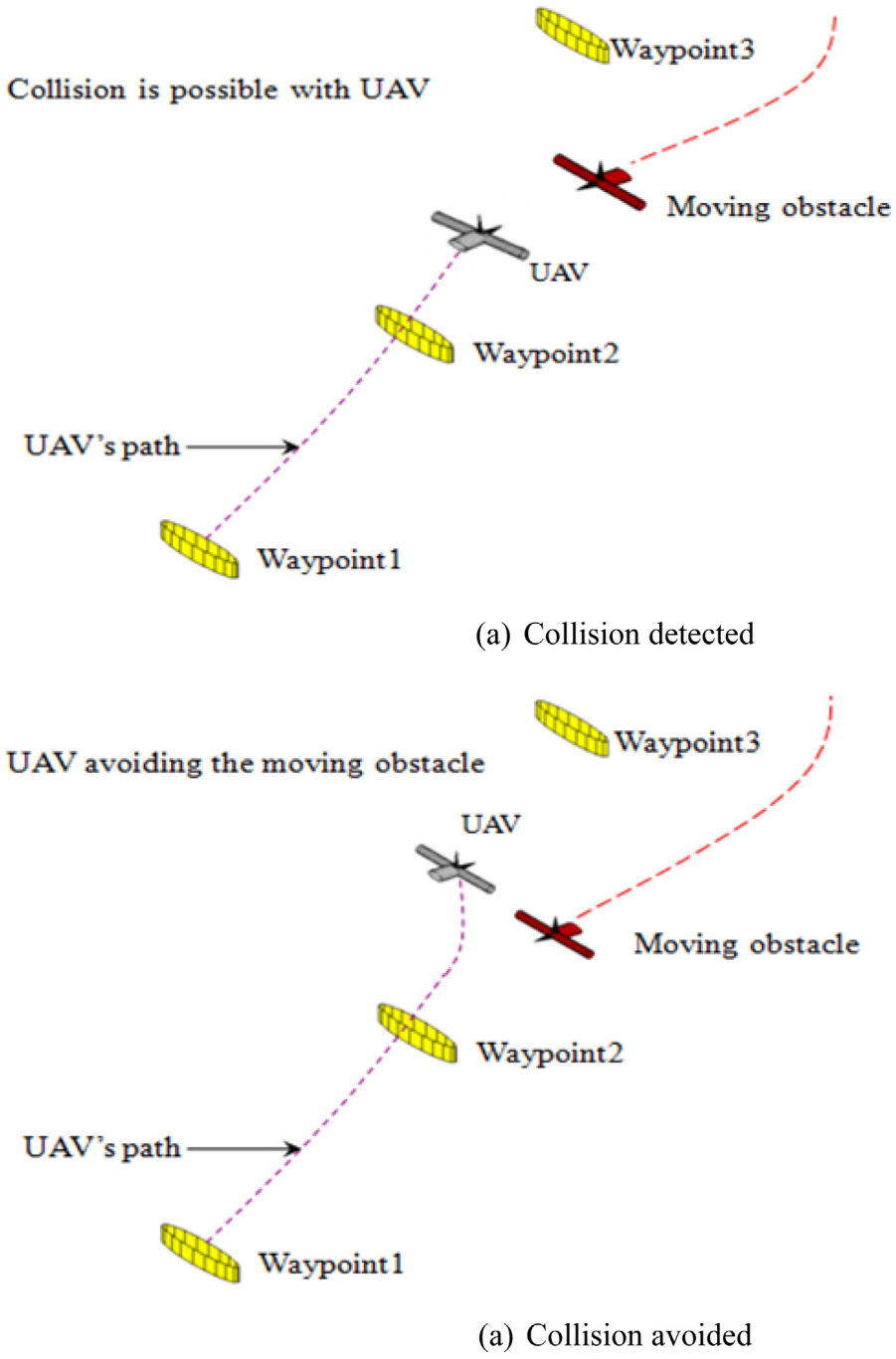


Fig. 27. UAV avoiding moving obstacle along its path.

the convergence to have one–one mapping, because the four different locations ($\pm x$, $\pm y$) map to the same solution as they have considered the mission waypoints limit inside the entire map boundaries. But our proposed algorithm has restricted waypoints search space with respect to the current UAV location and not the entire map boundaries. Hence, the phase angle encoding does not impact the result of the planner.

The four algorithms were compared based on the time taken for optimization process to find collision free alternate path and the length of optimized path chosen by them. The path chosen by the four algorithms are shown from Figs. 29–32. It is seen from the graph that SA had chosen 2 alternate waypoints to avoid one of the obstacle. The performance comparison is listed in Table 4.

The waypoints are same as in Section 5.1 and the position of obstacles is

obsx = [60000 135000 290400 290200 290000]
obsy = [7000 12000 17300 17400 17500]
obsz = [200 200 200 200 200]

The first obstacle is not along the path of UAV while the last 3 obstacles are very close and nearly overlap with one another. Hence the avoidance algorithm has to avoid the second obstacle and the group of last 3 obstacles. The path length of UAV's travel without any obstacle is 431,630 metres.

The results of four planners in Table 4 show that the optimized PSO planner has identified a short path of travel compared to

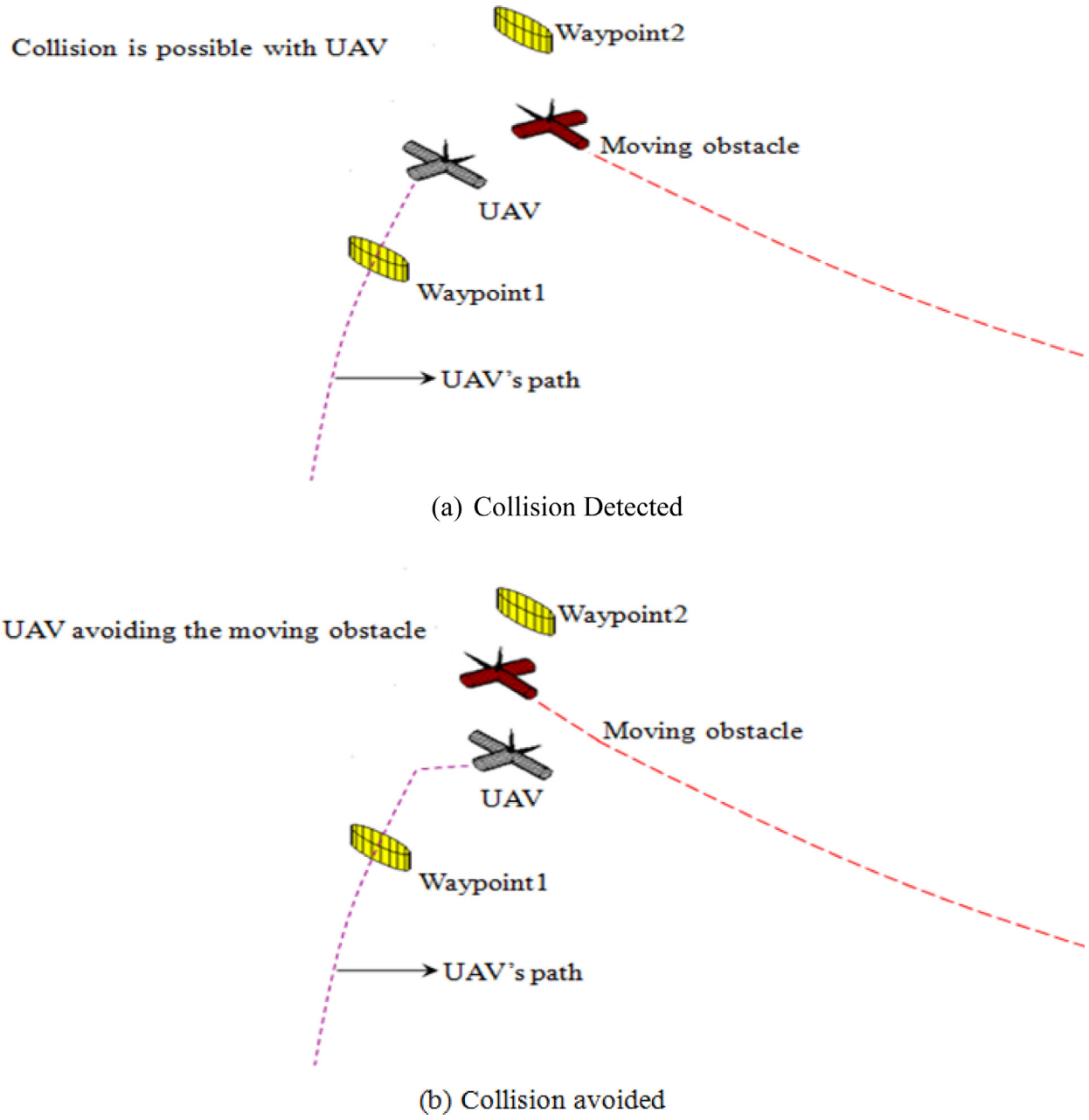


Fig. 28. UAV avoiding left moving intruder.

other planners. The numbers in bold indicate the new waypoints identified to avoid collision.

For further comparison of algorithms, the Statistical Front-Dominance Ranking Procedure (SFDRP) metric to measure the ability of the proposed path planning algorithm is used as reported in [3]. The comparison results are reported in Table 5. The representation in each cell of each table represents when a planner in the *Y*-axis is better (less dominated, in white), equivalent (not statistically different, in grey), or worse (more dominated, in black) than a planner in the *X*-axis [3,37].

In addition to these statistical graphical results, the planners are compared based on certain parameters namely number of successful runs out of total 10 runs (SR), Time of computation (TC), average of Convergence speed (CS) which represents the generation or iteration when all the constraints are satisfied, and overall path length ratio from source to destination (PLR). PLR is obtained by adding each segment length for the total trajectory divided by the straight-line distance between source and destination. PLR should be less than 1.5 for a good alternate trajectory. The above metrics are averaged with respect to the successful runs and if the objectives/constraints of any planner is not satisfied through all 10 runs, the corresponding entry of the metric will be noted as N/A. The results are shown in Table 6 for 40,60,80,100 obstacle scenario. The results of obstacles less than 40 are not reported here since all the three planners performed well. As the performance of the reported planners started degrading only with a greater number of obstacles, only those results are reported for analysis. PSO consumes less TC than GAs, since it is computationally less expensive as there is no population. SA takes less computation time compared to PSO but takes more iteration to converge. But finally, PSO scores more than other two algorithms in producing successful output by finding out alternate trajectory satisfying the required constraints. It is because PSO combines the advantage of both GA and

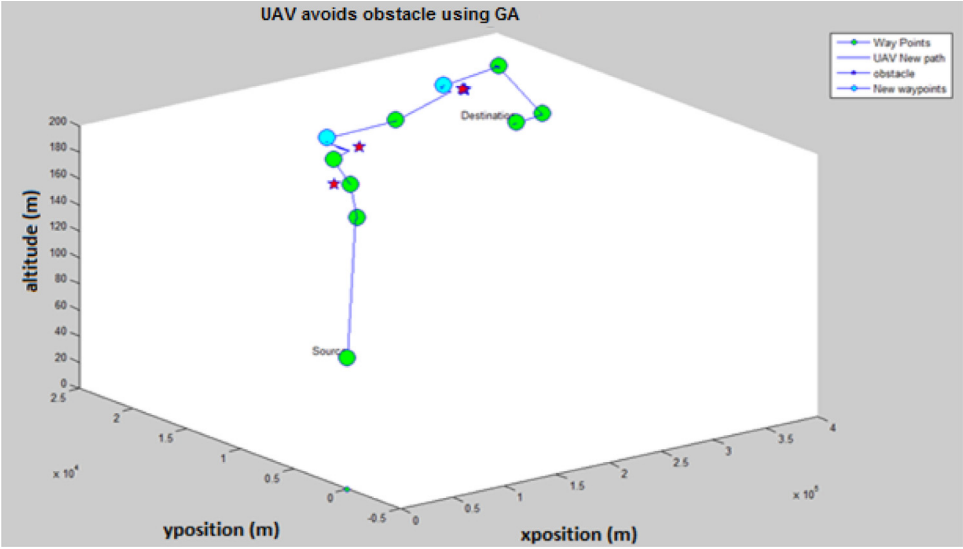


Fig. 29. Optimized path generated by GA.

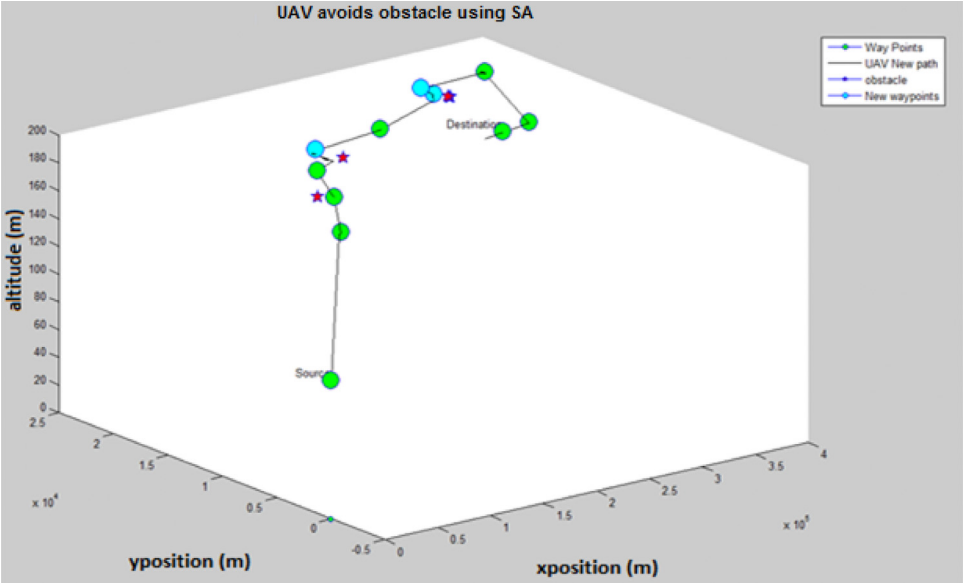


Fig. 30. Optimized path generated by SA.

Table 4
Comparison of planners.

Algorithm	New waypoints identified	Optimized travel path length (m)	Algorithm running time (s)
GA	x=[0 20000 70000 100000 **131190** 200000 **285710** 350000 300000 270000] y=[0 1000 6480 11000 **14672** 15000 **18871** 20000 11000 10500]	431780	32.0345
SA	x=[0 20000 70000 100000 **130990** 200000 **285510** **287430** 350000 300000 270000] y=[0 1000 6500 11000 **14172** 15000 **18371** **19749** 20000 11000 10500]	432150	26.7969
θ-MAFOA	x=[0 20000 70000 100000 **131280** 200000 **284710** 350000 300000 270000] y=[0 1000 6510 11000 **14188** 15000 **18361** 20000 11000 10500]	431720	29.0156
PSO	x=[0 20000 70000 100000 **130190** 200000 **284710** 350000 300000 270000] y=[0 1000 6510 11000 **14153** 15000 **18371** 20000 11000 10500]	431730	28.0938

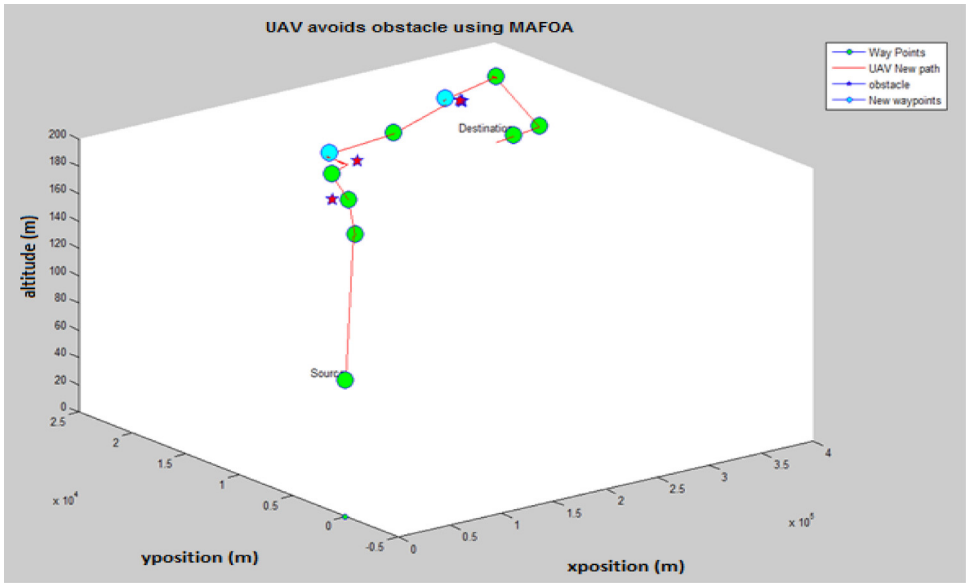


Fig. 31. Optimized path generated by θ-MAFOA.

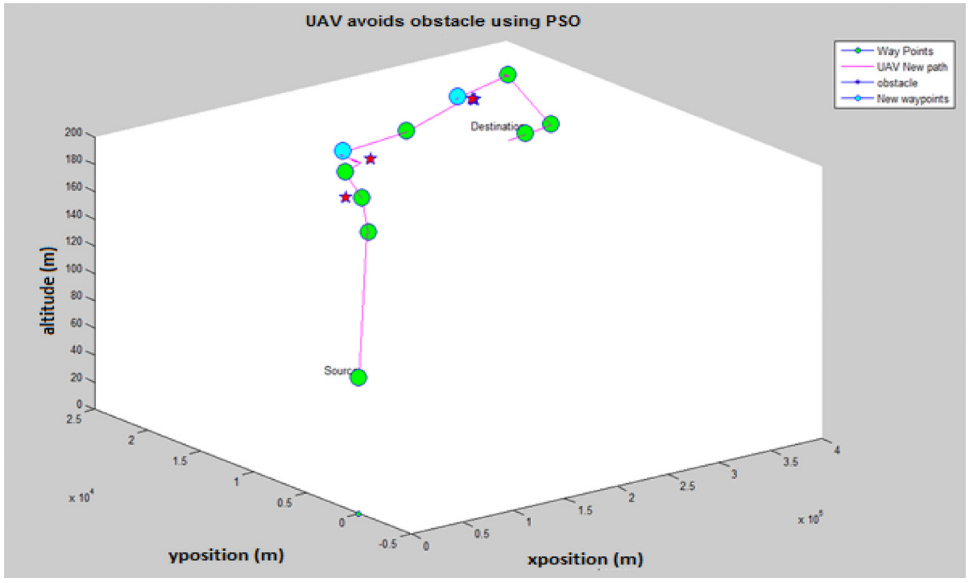


Fig. 32. Optimized path generated by PSO.

SA. The population-based search and less expensive computation makes PSO more suitable for real time trajectory path planning of UAV. The performance of θ-MAFOA algorithm is similar to that of our proposed PSO-CA algorithm in the successful generation of optimized path. Mutation adaptation mechanism of θ-MAFOA slightly improves the searching ability but phase angle added with location to increase convergence is of no use in our proposed algorithm except consuming more time. So, it is to be concluded that the powerful collision avoidance algorithm designed with the basic PSO is better than all other planners in terms of producing short and safe path in less time.

The performance of the proposed algorithm is studied with two other variants of PSO namely Multi-start Particle Swarm Optimizer (MPSO) and Guaranteed Convergence PSO (GCPSO) reported by Bergh and Engelbrecht in [46]. A solution to the problem of stagnation in original PSO is to change the velocity update equation for the global best particle to force a position change of the global best as in [46] which is GCPSO. Multi start PSO (slope) re-initialized the swarm whenever the change in fitness value remains constant continuously for 5 iterations. The mean and median values of fitness function for the two PSO variants are compared with that of original PSO and reported in Table 7. The results are reported for 4 obstacle case shown in Fig. 17 with waypoints as in Section 5.1. The results show that there is not much difference in the performance of the three planners.

7. Conclusion

In this work, the dynamic trajectory adjustment when sudden threats pop-up along the path of UAV is considered and implemented for Six Degrees of Freedom UAV modelled in Matlab. The designed UAV was allowed to move in predefined trajectory and the proposed PSO-CA algorithm for collision avoidance was tested after inducing simulated threats along its path. Both stationary and other UAVs modelled using 6-DOF appeared along its path

Table 5
Statistical analysis of 4 planners in scenario with 40,60,80,100 obstacles.

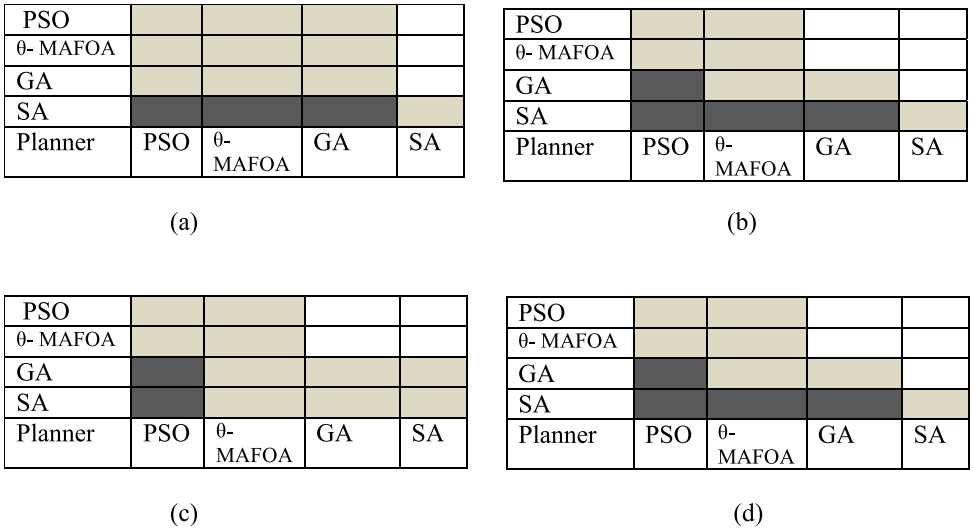


Table 6
Comparison of planners.

(a)					(b)				
40 Obstacle scenario					60 Obstacle scenario				
Planner	PLR	TC (s)	CS	SR	Planner	PLR	TC	CS	SR
PSO	1.28	62.03	5.24	10	PSO	1.37	69.48	7.53	10
θ-MAFOA	1.28	65.21	5.24	10	θ-MAFOA	1.39	72.21	7.57	10
GA	1.44	74.62	5.84	10	GA	1.52	79.02	7.62	9
SA	1.46	56.22	9.43	9	SA	1.82	64.32	10.20	8
(c)					(d)				
80 Obstacle scenario					100 Obstacle scenario				
Planner	PLR	TC (s)	CS	SR	Planner	PLR	TC	CS	SR
PSO	1.40	95.94	10.56	9	PSO	1.49	117.35	22.16	9
θ-MAFOA	1.42	98.42	10.51	9	θ-MAFOA	1.49	124.76	22.17	9
GA	1.62	107.21	11.27	8	GA	2.02	132.11	25.23	7
SA	1.91	88.16	12.24	8	SA	2.23	99.18	32.14	6

Table 7
Comparison of variants of PSO.

Algorithm	Mean fitness f	Median fitness f
Avoid obstacle 2 at location (135000,12000,200)		
PSO	1.0110e+05	1.0117e+05
GCPSO	1.0110e+05	1.0017e+05
MPSO	1.0080e+05	1.0095e+05
Avoid 3 continuous obstacles (290400, 17300,200), (290200, 17400,200), (290000, 17500,200)		
PSO	1.5119e+05	1.5809e+05
GCPSO	1.5099e+05	1.5024e+05
MPSO	1.5097e+05	1.5102e+05

and the UAV successfully avoided the obstacles by finding alternate safe path. The UAV detects collision and identifies alternate optimal trajectory when an unknown sudden threat comes under radar view which is the main feature of this work. The robustness of the algorithm was demonstrated by increasing the number of obstacles and the UAV being successfully steered along safest path in many cases. It is concluded from the graphical results that it is functional in an obstacle-rich environment and could be implemented in all UAVs for real time applications making them fly without any manual intervention. The proposed algorithm can easily be extended for cooperative UAV formation flying with collision avoidance.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106168.

Acknowledgements

The authors acknowledge the support extended by Defense Research and Development Organisation (AR&DB, SIGMA Panel), India through financial grants ARDB01/2021807M/I (sanction code: MSRB (ARDB)/TM/ARDB/GIA/16-17/0030). The authors would like

Table A.1
Fixed design parameters of UAV.

Description	Parameters
Mass of an aircraft, area	Mass=1000 kg, refarea=170 m^2
Span, chord, product of inertia	b=75 m, c=2.46 m, $I_{xz} = 962$ kg m^2
Moment of inertia	$I_{xx} = 66389$ kg m^2, $I_{yy} = 4820$ kg m^2, $I_{zz} = 70544$ kg m^2
Density	$\rho = 1.1$ kg/m^3
elevator trim	2.75/57.3
Thrust	100*9.81 N

Table A.2
Initial Parameters of UAV.

Initial parameters for UAV modelling			
Parameters	Value	Parameters	Value
X Position	0 m	Elevator command	0
Y Position	0 m	Aileron command	0
Z Position	100 m	Pitch Angle	0.069 degrees
Velocity along X axis	15 m/s	Roll Angle	0
Velocity along Y axis	0 m/s	Angle of Attack	0.069 degrees
Velocity along Z axis	0 m/s	Side slip Angle	0
Angular Rate (p,q,r)	0	Reference speed	15 m/s
Roll Command	0.1745	Cross track error	0

to thank all SIGMA panel members for their suggestions and Professor (retired) Dr. K. Sudhakar, IIT Bombay for his valuable reviews. The authors also submit sincere thanks for the anonymous reviewers of this journal.

Appendix. 6-DOF UAV Model

See Tables A.1 and A.2.

WAYPOINT LIST AND POSITION OF OBSTACLES

Waypoints
x = [0 20000 70000 100000 200000 350000 300000 270000];
y = [0 1000 6500 11000 15000 20000 11000 10500];
z = [100 200 200 200 200 200 200 200];

Position of obstacles
Altitude or Z coordinates of obstacle set constant as 200

Obstacle positions for Fig. 17
obsx=[60000 160000 240000 300000];
obsy=[7000 13000 16100 16000];

Obstacle positions for Fig. 19
obsx=[60000 135000 160000 170000 230000 229000 270000 267500 265000 262500 260000 257500 255000 252500 250000 247500 245000 242500 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 240000 242500 245000 247500 250000 252500 253750 255000 257500 260000 262500 265000 267500 270000 242500 245000 247500 250000 252500 255000 257500 300100];
obsy=[7000 12000 13000 12500 16000 15700 17400 17400 17400 17400 17400 17400 17400 17400 17400 17400 17400 17400 17400 17400 17400 17300 17200 17100 17000 16900 16800 16700 16600 16500 16400 16300 16200 16100 16000 15900 15600 15300 15000 14700 14400 14250 14800 15200 15600 16000 16400 16800 17200 16100 16300 16500 16700 16900 17100 17300 10400];

Obstacle positions for Fig. 20
obsx=[300000];
obsy=[18000];

Obstacle positions for Fig. 21
obsx=1.0e+05 *[0.2393 1.1433 1.4555 2.0771];
obsy=1.0e+04 *[0.2485 1.2438 1.4095 1.1229];

Obstacle positions for Fig. 22
obsx=1.0e+05 *[0.2393 1.1433 1.1000 1.4555 2.0771];
obsy=1.0e+04 *[0.2485 1.2438 0.9500 1.4095 1.1229];

Obstacle Positions for Fig. 23
obsx=[300100]; obsy=[10400];

MOVING AND STATIONARY OBSTACLE AVOIDANCE (video data, Figs. 27 and 28)

Waypoints for UAV
x=[0 20000 30000 40000 60000 75000];
y=[0 10000 14000 17000 20000 23000];
z=[100 200 200 200 200 200];

Waypoints for moving obstacle 1
x1=[50000 20000];
y1=[18000 10000];

Waypoints for moving obstacle 2
x2=[110000 44500];
y2=[25000 18000];

Stationary Obstacle
obsxx=[13000 36000 68000];
obsyy=[6000 15000 19000];

References

[1] Matthew T. DeGarmo, Issues Concerning Integration of Unmanned Aerial Vehicles in Civil Airspace, Center for Advanced Aviation System Development, McLean, Virginia, 2004.
[2] Naifeng Wen, Lingling Zhao, Xiaohong Su, Peijun Ma, UAV online path planning algorithm in a low altitude dangerous environment, IEEE/CAA J. Autom. Sin. 2 (2) (2015).
[3] Peng Yang, Ke Tang, Jose A. Lozano, Xianbin Cao, Path planning for single unmanned aerial vehicle by separately evolving waypoints, IEEE Trans. Robot. 31 (5) (2015).
[4] Yucong Lin, Srikanth Saripalli, Sampling-based path planning for UAV collision avoidance, IEEE Trans. Intell. Transp. Syst. 18 (11) (2017).
[5] Yangguang Fu, Mingyue Ding, Chengping Zhou, Hu Hanping, Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization, IEEE Trans. Syst. Man Cybern: Syst. 43 (6) (2013).
[6] X.Y. Zhang, H.B. Duan, Q.N. Luo, Levenberg–Marquardt based artificial physics method for mobile robot oscillation alleviation, Sci. China Phys. Mech. Astron. 57 (9) (2014) 1771–1777.
[7] S. Bayilia, F. Polatb, Limited-Damage A*: A path search algorithm that considers damage as a feasibility criterion, Knowl.-Based Syst. 24 (4) (2011) 501–512.
[8] H. Vazquez-Leal, A. Marin-Hernandez, Y. Khan, et al., Exploring collision-free path planning by using homotopy continuation methods, Appl. Math. Comput. 219 (2013) 7514–7532.

[9] T.T. Mac, C. Copot, D.T. Tran, R.D. Keyser, Heuristic approaches in robot path planning: A survey, Robot. Auton. Syst. 86 (2016) 13–28.
[10] E. Besada-Portas, L. de la Torre, J.M. de la Cruz, B. de Andres-Toro, Evolutionary trajectory planner for multiple UAVs in realistic scenarios, IEEE Trans. Robot. 26 (4) (2010) 619–634.
[11] G.G. Wang, H.E. Chu, S. Mirjalili, Three-dimensional path planning for UCAV using an improved bat algorithm, Aerosp. Sci. Technol. 49 (2016) 231–238.
[12] V. Roberge, M. Tarbouchi, G. Labonté, Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning, IEEE Trans. Ind. Inform. 9 (1) (2013) 132–141.
[13] Y.G. Fu, M.Y. Ding, C.P. Zhou, Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV, IEEE Trans. Syst. Man Cybern. A: Syst. Hum. 42 (2) (2012) 511–526.
[14] X. Wu, et al., A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs, Appl. Soft Comput. J. (2018) http://dx.doi.org/10.1016/j.asoc.2018.09.011.
[15] H.B. Duan, X.Y. Zhang, J. Wu, G.J. Ma, Max–min adaptive ant colony optimization approach to multi-UAVs coordinated trajectory replanning in dynamic and uncertain environments, J. Bionic Eng. 6 (2) (2009) 161–173.
[16] C.F. Xu, H.B. Duan, F. Liu, Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning, Aerosp. Sci. Technol. 14 (8) (2010) 535–541.
[17] Y. Chen, J. Yu, Y. Mei, Y. Wang, X. Su, Modified central force optimization (MCFO) algorithm for 3D UAV path planning, Neurocomputing 171 (2016) 878–888.
[18] X.Y. Zhang, H.B. Duan, An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning, Appl. Soft Comput. 26 (2015) 270–284.
[19] H.B. Duan, L.Z. Huang, Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning, Neurocomputing 125 (2014) 166–171.
[20] A.A. Khaled, S. Hosseini, Fuzzy adaptive imperialist competitive algorithm for global optimization, Neural Comput. Appl. 26 (4) (2015) 813–825.
[21] Xiangyin Zhang, Xingyang Lu, Songmin Jia, Xiuzhi Li, A novel phase angle-encoded fruit fly optimization algorithm with mutation adaptation mechanism applied to UAV path planning, Appl. Soft Comput. J. https://doi.org/10.1016/j.asoc.2018.05.030.
[22] P.K. Das, H.S. Behera, B.K. Panigrahi, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning, Swarm Evol. Comput. 28 (2016) 14–28.
[23] Y.G. Fu, M.Y. Ding, C.P. Zhou, H.P. Hu, Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization, IEEE Trans. Syst. Man Cybern. Syst. 43 (6) (2013) 1451–1465.
[24] Domenico Accardo, Giancarmine Fasano, Lidia Forlenza, Antonio Moccia, Attilio Rispoli, Flight test of a radar-based tracking system for UAS sense and avoid, IEEE Trans. Aerosp. Electron. Syst. 49 (2) (2013).
[25] Fethi Belkhouche, Modeling and Calculating the collision risk for air vehicles, IEEE Trans. Veh. Technol. 62 (5) (2013).
[26] Chunbo Luo, Sally I. McClean, Gerard Parr, Luke Teacy, Renzo De Nardi, UAV position estimation and collision avoidance using the extended kalman filter, IEEE Trans. Veh. Technol. 62 (6) (2013).
[27] Yao Peng, Wang Honglun, Su Zikang, UAV feasible path planning based on disturbed fluid and trajectory propagation, Chin. J. Aeronaut. 28 (4) (2015) 1163–1177.
[28] Peng Yao, Honglun Wang, Zikang Su, Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment, Aerosp. Sci. Technol. 47 (2015) 269–279.
[29] Yang b. Liua, Xuejun Zhanga, Xiangmin Guanc, Daniel Delahayed, Delahayed adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization, Aerosp. Sci. Technol. 58 (2016) 92–102.
[30] Tiago Oliveira, A. Pedro Aguiar, Pedro Encarnaç, Moving path following for unmanned aerial vehicles with applications to single and multiple target tracking problems, IEEE Trans. Robot 32 (5) (2016).
[31] Youdan Kim, Seungkeun Kim, Antonios Tsourdos, Collision avoidance strategies for unmanned aerial vehicles in formation flight, IEEE Trans. Aerosp. Electron. Syst. 53 (6) (2017).
[32] Walton Pereira Coutinho, Maria Battarrab, Jörg Fliege, The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review, Comput. Ind. Eng. 120 (2018) 116–128.
[33] I. Hasircioglu, H.R. Topcuoglu, M. Ermis, 3-d path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms, in: Proc. Genet. Evol. Comput. Conf. 2008 pp. 1499–1506.
[34] I.K. Nikolos, N.C. Tsourveloudis, K.P. Valavanis, Volutionary algorithm based path planning for multiple UAV cooperation, in: Advances in Unmanned Aerial Vehicles, Springer-Verlag, Berlin, Germany, 2007, pp. 309–340.
[35] R. Zhang, C. Zheng, P. Yan, Route planning for unmanned air vehicles with multiple missions using an evolutionary algorithm, in: Proc. IEEE 3rd Int. Conf. Nat. Comput. 2007, pp. 1499–1506.
[36] C. Zheng, L. Li, F. Xu, F. Sun, M. Ding, Evolutionary route planner for unmanned air vehicles, IEEE Trans. Robot. 21 (4) (2005) 609–620.
[37] E. Besada-Portas, L. De La Torre, A. Moreno, J.L. Risco-Martín, On the performance comparison of multi-objective evolutionary UAV path planners, Inform. Sci. 238 (2013) 111–125.
[38] A.K. Marsh, Avoiding Midair Collisions. Aircraft Owners and Pilots Association. [Online] http://flighttraining.aopa.org/students/presolo/skills/midair.html.
[39] John H. Blakelock, Longitudinal dynamics, in: Automatic Control of Aircraft and Missiles, second ed., A Wiley Interscience publication, USA, 1991, pp. 7–142.
[40] Jan Roskam, Equations of motion and Axis systems, in: Airplane Flight Dynamics and Automatic Flight Controls, second ed., Design, Analysis and Research corporation, USA, 1998, pp. 3–34.
[41] M.V. Cook, Systems of axes and notations, in: Flight Dynamics Principles, second ed., Elsevier Ltd, 2007, pp. 12–30.
[42] David A. Caughey, Introduction to aircraft stability and control, in: Sibley School of Mechanical & Aerospace Engineering, Cornell University, Newyork, 2011.
[43] Y.V. Pehlivanoglu, O. Baysal, A. Hacioglu, Vibrational genetic algorithm based path planner for autonomous UAV in spatial data based environments, in: Proc. 3rd Int. Conf. Recent Adv. Space Technol. vol. 7 2007, pp. 573–578.
[44] A. Ratnaweera, S. Halgamuge, H. Watson, Particle swarm optimization with time varying acceleration coefficients, in: Proceedings of the International Conference on Soft Computing and Intelligent Systems, Coimbatore, India, July 26–28 2002, pp. 240–255.
[45] Kennedy J., R. Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, HI, USA, 12–17 May 2002.
[46] F. van den Bergh, A.P. Engelbrecht, A convergence proof for the particle swarm optimiser, Fund. Inform. 105 (4) (2010) 341–374.