Hindawi
International Journal of Aerospace Engineering
Volume 2020, Article ID 8820284, 17 pages
https://doi.org/10.1155/2020/8820284

Research Article

UAV Path Planning under Dynamic Threats Using an Improved PSO Algorithm

Jong-Jin Shin and Hyochoong Bang

Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea

Correspondence should be addressed to Hyochoong Bang; hcbang@ascl.kaist.ac.kr

Received 27 August 2020; Revised 20 November 2020; Accepted 8 December 2020; Published 31 December 2020

Academic Editor: Joseph Morlier

This paper presents the method to solve the problem of path planning for an unmanned aerial vehicle (UAV) in adversarial environments including radar-guided surface-to-air missiles (SAMs) and unknown threats. SAM lethal envelope and radar detection for SAM threats and line-of-sight (LOS) calculation for unknown threats are considered to compute the cost for path planning. In particular, dynamic SAM lethal envelope is taken into account for path planning in that SAM lethal envelope does change its direction according to the flight direction of UAV. In addition, terrain masking, nonisotropic radar cross section (RCS), and dynamic constraints of UAV are considered to determine the cost of the path. An improved particle swarm optimization (PSO) algorithm is proposed for finding an optimal path. The proposed algorithm is composed of preprocessing steps, multi-swarm PSO algorithm, and postprocessing steps. The Voronoi diagram and Dijkstra algorithm as preprocessing steps provide the initial path for the multi-swarm PSO algorithm which uses multiple swarms with sub-swarms for the balance between exploration and exploitation. Postprocessing steps include waypoint insertion and 3D path smoothing. The computation time is reduced by using the map generation, the coordinate transformation, and the graphic processing unit (GPU) implementation of the algorithm. Various simulations are carried out to compare the performance of the proposed method according to the number of iterations, the number of swarms, and the number of cost evaluation points. The t-test results show that the suggested method is statistically better than existing methods.

1. Introduction

In this paper, we propose the method to solve the problem of path planning for an unmanned aerial vehicle (UAV) in adversarial environments where there are radar-guided surface-to-air missiles (SAMs). The problem needs realistic models which take many aspects into consideration related to the hostile environments. They include dynamic SAM lethal envelope, radar detection, and unknown threat. In particular, we take into account terrain masking, nonisotropic radar cross section (RCS), and dynamic constraints of UAV when computing the cost for those hostile environments. There are few papers integrating all these aspects together. This paper suggests the improved particle swarm optimization (PSO) algorithm to offer an optimal path which can take these realistic models into consideration.

These days, UAVs are widely used in broad areas such as commercial and military applications [1, 2]. Path planning is definitely one of the main technologies that UAVs require [3]. The objective in UAV path planning is to complete a given mission while maximizing the safety of UAV until the mission is completed [4]. In military applications, there exist a lot of threats in route of the UAV. These threats include a SAM, a radar, and so on. Thus, the UAV path planning requires the route for avoiding these hostile threats to arrive at the target safely.

As a related field, robot path planning is for finding an optimal path to avoid a collision. This obstacle avoidance is the main purpose for the robot path planning. UAV path planning shares much in common since threats can be treated as obstacles. The similarity in both areas can make the path planning algorithms in robot applications applicable

to UAV path planning. However, there is a main difference between robot path planning and UAV path planning in that UAV path planning needs vehicle dynamic constraints [5]. UAV must maintain a certain velocity while flying. It means that sharp turns are not allowed because of dynamic constraints.

Although the current literature on UAV path planning does not consider the realistic models altogether, there are various papers which take each or several aspects into account. Terrain masking which is important for the survivability of UAV is considered in References [6, 7]. Nonisotropic RCS is taken into account in References [8–11]. Numerous path planning algorithms have been proposed. One category of path planning algorithms is graph-based method. This method include the A* and D* lite algorithm [12–15] and Voronoi diagram method [16, 17]. Metaheuristic optimization methods are another path planning algorithms. Among them, simulated annealing [18], ant colony [6], genetic algorithm [19–24], and PSO [3, 20, 22, 25, 26] algorithms have been widely used for path planning. In terms of implementing dynamic constraints of UAV, GA and PSO algorithms are preferable to the graph-based algorithms.

This paper deals with the path planning problem for UAV under the hostile environments of radar-guided SAMs. It is distinguished from the current literature in that dynamic SAM lethal envelope is considered in addition to radar detection and terrain masking is taken into consideration for computing the cost of radar detection and unknown threats. Dynamic SAM lethal envelope means that SAM lethal envelope changes its direction according to the flight direction of UAV. No previous works have dealt with dynamic SAM lethal envelope in calculating the threat cost of the SAM. We also take terrain masking into consideration. When UAV is under adversarial environments, UAV needs to fly as low as possible to minimize the detection by a radar. In that case, terrain masking is an important factor to increase the survivability of UAV. Terrain masking is considered in two kinds of calculations. One is to determine whether a radar can detect UAV and the other is line-of-sight (LOS) calculation to consider the unknown threats. Unknown threats may exist somewhere in the UAV route. If UAV is visible on the ground or UAV is not obscured by terrain, it may be shot down by those unknown threats. Therefore, the more visible UAV is around its location, the riskier UAV is at that location. In addition to dynamic SAM lethal envelope and terrain masking, the nonisotropic RCS of UAV is considered in calculating radar detectability. RCS changes dramatically depending on the attitudes of UAV. RCS as well as terrain masking affects the cost for radar detection.

In this paper, we propose the path planning method using an improved PSO algorithm, which is composed of preprocessing steps, multi-swarm PSO algorithm, and postprocessing steps. As preprocessing steps, the Voronoi diagram and Dijkstra algorithm are utilized to generate the initial path. Preprocessing steps are particularly effective for test cases where it is difficult to find an optimal path. The multi-swarm PSO algorithm use multiple swarms with a lot larger population size than traditional PSO algorithms thanks to graphic processing unit (GPU) implementation so that fast convergence is possible with a balance between exploration and exploitation. Various maps such as the flight altitude map, the terrain masking map, and the 3D LOS map are constructed, and the coordinate transformation is carried out to reduce the computation time. Postprocessing steps include the waypoint insertion and the path smoothing for decreasing the vulnerability due to the threats by inserting a waypoint. The results for various test cases are analyzed and compared with those of other PSO algorithms.

The remainder of this paper is as follows. UAV and threat modeling is described in Section 2. The path planning method is proposed in Section 3. In Section 4, the simulation parameters, results, and discussions in comparison to those of the existing methods are described. Finally, the conclusions are provided in Section 5.

2. UAV and Threat Modeling

2.1. UAV Modeling. The level flight with a constant velocity between waypoints is assumed for UAV. The waypoints are used to construct a simplified path of UAV. They are the reference points which provide the position information for UAV. The altitude of UAV is maintained between the neighboring waypoints. When the horizontal path is determined, the altitude of UAV is determined according to the horizontal path. Safety margin is also considered to avoid ground collision when determining the altitude of UAV. Dynamic constraints of UAV are implemented using the minimum turning radius ($r_{\min}$) and the turning angle (λ_t). The minimum turning radius is determined by the flight characteristics. The turning angle represents the angle between the two adjacent segments along the path of UAV. Sharp turns can be controlled using the turning angle as a dynamic constraint of UAV. Figure 1 shows the coordinates, path, and dynamic constraints of UAV mentioned above.

The coordinate transformation is carried out to reduce the computation time for the PSO algorithm as shown in Figure 2 [3, 27]. In a new coordinate system, the x' axis lies in a line from a starting point (S) to a destination (F) and the y' axis is perpendicular to the x' axis. A straight line SF is uniformly divided into $(D+1)$ segments. D is the number of waypoints WP_k ($k = 1, 2, \cdots, D$). The length of equally distributed segments is defined as dx', and the coordinates of each waypoint can be represented in a new coordinate system $x'y'$ using the following equations.

$$\begin{cases} \mathrm{WP}_k = \left(k\,dx', y_k'\right), \\ dx' = \dfrac{\left|\overrightarrow{SF}\right|}{D+1}. \end{cases} \tag{1}$$

Then, the waypoints to be determined can be expressed as $(y_1', y_2', \cdots, y_D')$ in 1D rather than both x and y positions.

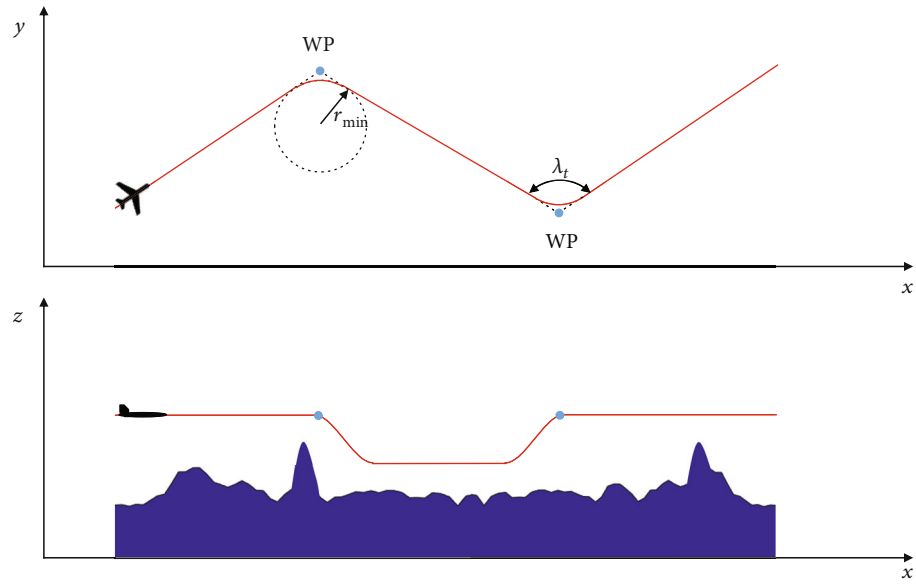


FIGURE 1: Coordinates, path, and dynamic constraints of UAV. WP denotes the waypoint. r_{min} and λ_t are the minimum turning radius and the turning angle, respectively.

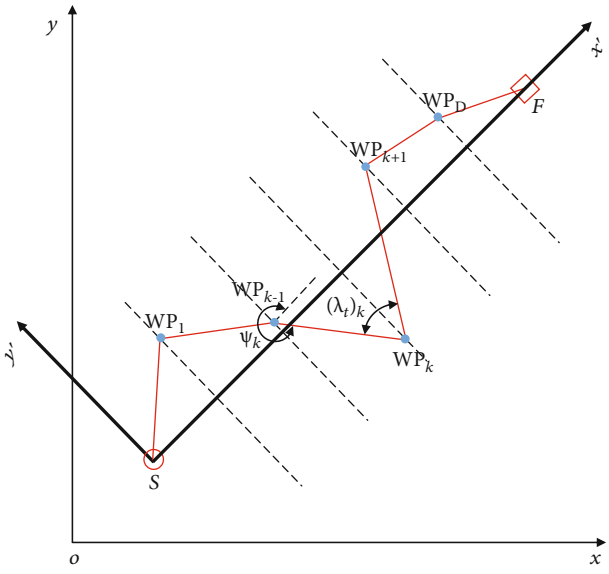


FIGURE 2: The coordinate transformation.

The segment length between two adjacent waypoints can be obtained using the following equation.

$$\left|\overrightarrow{WP_{k-1}WP_k}\right| = \sqrt{\left(dx'\right)^2 + \left(y'_k - y'_{k-1}\right)^2}, \quad (2)$$

where $k = 1, 2, \cdots, D+1$. WP_0 and WP_{D+1} become S and F, respectively. y'_0 and y'_{D+1} are y' positions of S and F. Therefore, $y'_0 = y'_{D+1} = 0$.

The heading angle (ψ) for $\overrightarrow{WP_{k-1}WP_k}$ in a new coordinate system can be expressed using the following equation.

$$\psi_k = \text{atan2}\left(y'_k - y'_{k-1}, dx'\right), \quad (3)$$

where $k = 1, 2, \cdots, D+1$. atan2 is the four-quadrant inverse tangent.

The turning angle can be expressed using the following equation.

$$(\lambda_t)_k = \min\left\{|\pi + (\psi_{k+1} - \psi_k)|, |\pi - (\psi_{k+1} - \psi_k)|\right\}, \quad (4)$$

where $k = 1, 2, \cdots, D$. $(\lambda_t)_k$ can have a value from 0 to π.

Nonisotropic RCS of UAV is modeled. RCS is dependent on the position and attitudes of UAV relative to the radar. RCS is used to calculate the radar detectability. RCS values used in this paper are discussed in detail in Section 2.2.

2.2. Threat Modeling. Threat modeling is composed of 3 parts. SAM lethal envelope and radar detection are two components for SAM threats. The other is the unknown threat of which location is not known when the path is planned.

SAM lethal envelope is the launch envelope which is drawn from calculating the probability of kill at the position of UAV at the time of SAM launch. SAM lethal envelope is assumed to be elliptical, and the SAM is located at a focus of the ellipse. For a stationary target, a circular shape can be used for SAM lethal envelope. For a fast moving target such as UAV, SAM lethal envelope as a launch envelope can be modeled as an ellipse. In this paper, dynamic SAM lethal envelope is taken into account in that the envelope changes its direction in terms of the heading of UAV. The probability of kill usually becomes lower when the offset of a UAV flight trajectory from the SAM site gets larger. One of the reasons for low probability of kill is that more maneuver of SAM is

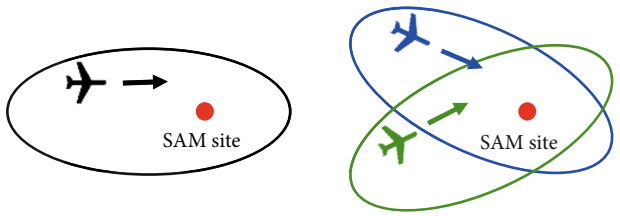


Figure 3: Dynamic SAM lethal envelope.

needed for a large offset. In Figure 3, we see that the semimajor axis of the elliptical envelope is parallel to the flight direction of UAV. If the flight direction of UAV is changed, the envelope also changes the direction while the location of the SAM is fixed at the original place. Once UAV is within the SAM lethal envelope, the cost of SAM engagement is added to the total cost. Of course, if UAV is out of the envelope, there is no cost induced from the envelope.

The locations of the SAM and the radar are assumed to be known and located at the same place. Radar detection signal is inversely proportional to the 4^{th} power of the distance between the radar and UAV. The altitude of UAV as well as the horizontal path is considered in calculating the cost of radar detection. Terrain masking is utilized to maximize the safety of UAV. Once the horizontal path is determined, the altitudes of UAV according to that horizontal path are determined automatically assuming the level flight with a constant velocity between two neighboring waypoints. This altitude is determined by the path planner considering the safety margin of UAV flight.

The cost for unknown threats is calculated from LOS calculation. When UAV is visible from the ground near UAV, it can be in danger due to possible threats. LOS calculations are conducted from the ground regions within a specified distance from UAV. The exposure density of UAV from the ground regions using LOS calculations is converted to the cost for unknown threats. Terrain masking is considered in both cost calculations of radar detection and LOS for unknown threats. When UAV flies through the valley, the cost for unknown threat will be low due to terrain masking.

Threat costs for traveling along the path include SAM lethal envelope, radar detection, and LOS calculation at the same time. Integration of the cost over the entire path would be needed for exact calculation of total cost. However, we

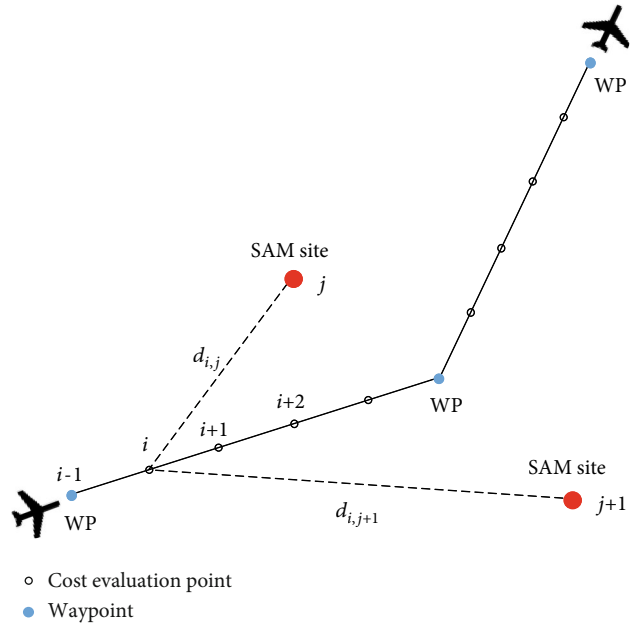


Figure 4: Cost calculation at cost evaluation points along a path.

compute the cost at cost evaluation points along an entire path and consider the distance between the points for calculating total costs. This approximation is used considering a long distance of UAV path and the computation time. The number of cost evaluation points in a segment is determined as a constant, and it is the same for all segments. Figure 4 shows two segments as a part of the path where i denotes the UAV position to calculate the threat cost and j stands for a SAM site. $d_{i-1,i}$ which is the distance between two adjacent cost evaluation points is the distance between UAV positions $i-1$ and i along a path. $d_{i,j}$ denotes the distance between UAV location i and the j^{th} radar.

The cost of total threats including SAM-related threats and unknown threats is the summation of three components when all the turning angles as dynamic constraints are greater than a threshold. Otherwise, the penalty cost (P_t) which is a very high value is given for the path which does not satisfy dynamic constraints. Total threat cost is calculated as a summation over N cost evaluation points and M SAM sites using the following equation.

$$J_{\text{threat}} = \begin{cases} \sum_{i=1}^{N} d_{i-1,i} \left\{ \sum_{j=1}^{M} \left(\text{SLE}_{i,j} + \dfrac{C_{\text{R}} \text{TM}_{i,j}\, \sigma_{i,j}}{d_{i,j}^{4}} \right) + \dfrac{C_{\text{L}}\, L_i}{L_t} \right\}, & \text{if all } \lambda_t \geq T_t, \\ P_t, & \text{otherwise,} \end{cases} \tag{5}$$

where $\text{SLE}_{i,j}$ is the cost of SAM lethal envelope at the UAV location i due to the j^{th} SAM. C_{R} is a constant for the cost of radar detection. $\text{TM}_{i,j}$ is the value for terrain masking. $\sigma_{i,j}$ denotes the RCS value of UAV. C_{L} is a constant for the cost of LOS calculation. L_i and L_t denote the number of ground positions where UAV is visible and the number of total ground positions where LOS calculations are conducted for the UAV location i, respectively. T_t denotes a threshold for the turning angle.

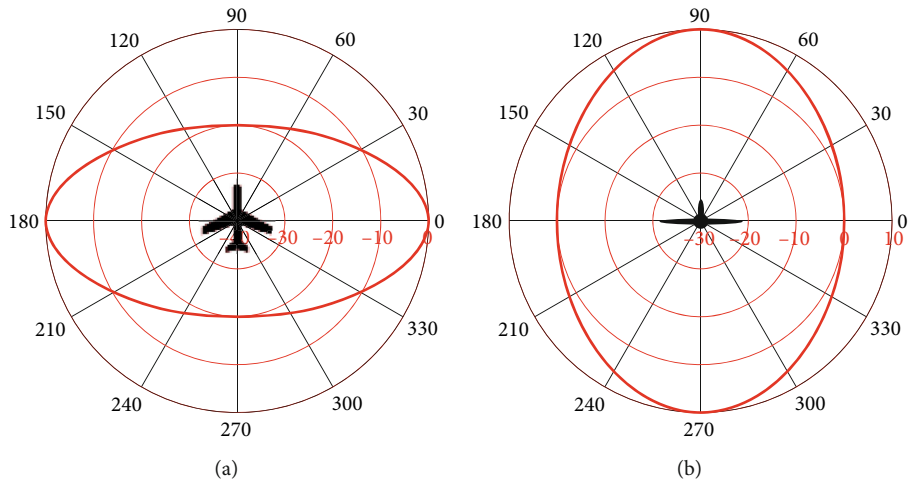


Figure 5: Horizontal RCS patterns (a) and vertical RCS patterns (b) for UAV. The RCS unit is in dBsm.

$\mathrm{TM}_{i,j}$ represents whether UAV is masked by the terrain from a radar. If UAV is not masked by the terrain, the value becomes 1 and the radar equation is applied for calculating the cost of radar detection. If UAV is masked, the value is 0. RCS values ($\sigma_{i,j}$) are calculated from widely accepted approximation for the ellipsoid as shown in the following equation [28]. The parameters ($a = 0.3172, b = 0.1784$, and $c = 1.003$) are selected in such a way that RCS is relatively small when viewed from the front or tail side but RCS is relatively large when viewed from the side or below [9]. Figure 5 shows RCS for the ellipsoid, which varies according to θ and ϕ. The positions of UAV relative to a radar and UAV heading are used for calculating RCS. A pitch and a bank angle of UAV are assumed to be zero.

$$\sigma_{i,j} = \frac{\pi a^2 b^2 c^2}{\left(a^2 (\sin\theta)^2 (\cos\phi)^2 + b^2 (\sin\theta)^2 (\sin\phi)^2 + c^2 (\cos\phi)^2\right)^2}, \tag{6}$$

where θ denotes the angle between UAV's heading and the direction to a receiving radar. ϕ denotes the angle between the wing axis of UAV and the projected line of a receiving radar on the plane of the wing and the tail axis.

The cost for unknown threats is calculated from LOS calculations from the ground region near UAV. LOS calculations are conducted at the ground positions within a specified distance around the position of UAV. When UAV is visible from all ground positions, $C_{\mathrm{L}} L_i / L_t$ becomes C_{L}, which is the maximum value for the cost of unknown threats.

In Figure 6, the total costs for two different UAV headings are shown. SAM lethal envelope changes its direction according to the heading of UAV. The cost of radar detection is computed considering terrain masking and RCS. To determine whether UAV is masked by the terrain for each radar, the terrain masking map is generated. The terrain masking map for radar detectability is generated for every radar, and it contains the maximum altitude of UAV below which UAV can be masked by the terrain for a certain radar. The costs of SAM lethal envelope and radar detection are computed once a path is generated since SAM lethal envelope changes its direction according to the heading of UAV. The cost of radar detection depends on the nonisotropic RCS of UAV, which changes according to the attitude of UAV. LOS calculations should be carried out for the altitudes as well as the horizontal positions of UAV. Figure 7 shows the 3D LOS map for several different altitudes and a stack of LOS maps which are used to compute the vulnerability of UAV for unknown threats. UAV with lower altitude is less vulnerable to unknown threats as shown in Figure 7. Safer areas are represented in blue colors. The terrain masking map and the 3D LOS map which take long time for calculation are precomputed for the region of interest.

3. Path Planning Method

3.1. The Traditional and Standard PSO Algorithms. PSO is a population-based stochastic algorithm which is inspired by social behavior of birds and fishes [29, 30]. It combines self-experience with social experience. The particles are randomly initialized at first. Then, they search for the solution space based on the position and the velocity update. Each particle i has a position and a velocity.

$$\begin{cases} \mathbf{y}_i = (y_{i1}, y_{i2}, \cdots, y_{iD}), \\ \mathbf{v}_i = (v_{i1}, v_{i2}, \cdots, v_{iD}), \end{cases} \tag{7}$$

where D denotes the dimension of the particle, which is the number of variables. $\mathbf{y}_i$ is a set of waypoint positions.

Each particle keeps track of individual best and global best of all particles. **pbest** and **gbest** denote the individual best and global best vector, respectively.

$$\begin{cases} \mathbf{pbest}_i = (\mathrm{pbest}_{i1}, \mathrm{pbest}_{i2}, \cdots, \mathrm{pbest}_{iD}), \\ \mathbf{gbest} = (\mathrm{gbest}_1, \mathrm{gbest}_2, \cdots, \mathrm{gbest}_D). \end{cases} \tag{8}$$

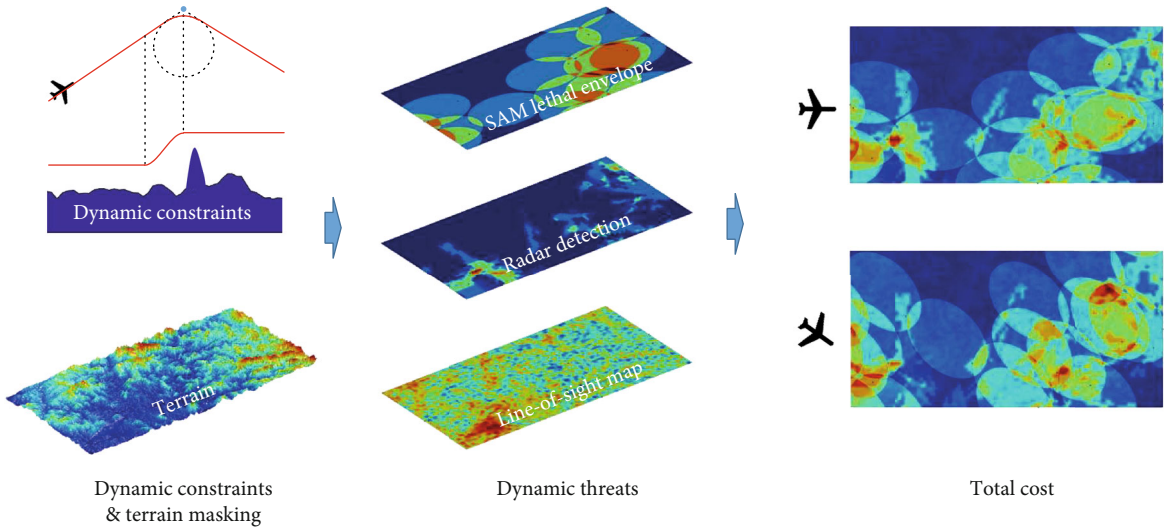


Figure 6: Total cost considering the dynamic threats.

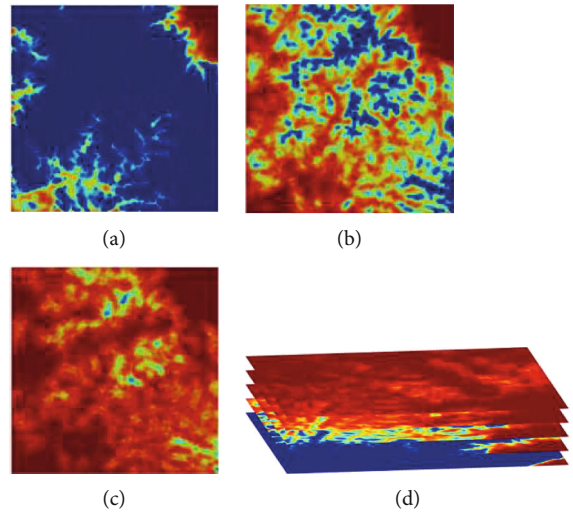


Figure 7: The LOS map according to the altitudes: (a) the altitude of 500 m; (b) the altitude of 1000 m; (c) the altitude of 1500 m; (d) a stack of LOS maps used for the LOS calculations.

Fitness functions are evaluated for each particle, and **pbest** and **gbest** are determined after an iteration. Then, the velocity and the position are updated according to the following equations.

$$\begin{cases} \mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + c_1\ \mathbf{rand}_1^k .* \left(\mathbf{pbest}_i^k - \mathbf{y}_i^k\right) + c_2\ \mathbf{rand}_2^k .* \left(\mathbf{gbest}^k - \mathbf{y}_i^k\right), \\ \mathbf{y}_i^{k+1} = \mathbf{y}_i^k + \mathbf{v}_i^{k+1}, \end{cases} \tag{9}$$

where k is the iteration number. w, c_1, and c_2 denote the inertia weight, the individual learning coefficient, and the global learning coefficient, respectively. $\mathbf{rand}_1$ and $\mathbf{rand}_2$ are vectors of random values which range from 0 to 1.

Fitness function evaluations and the update of the velocity and the position are iterated until the criteria are met. The above traditional PSO algorithm can be implemented easily, and it needs a few parameters. It can be utilized for solving the problem of UAV path planning. However, it may be possible that the algorithm converges prematurely or falls into local optima.

In contrast to the traditional PSO algorithm, the standard PSO algorithm was proposed in [31]. Among some differences between these algorithms, there are two

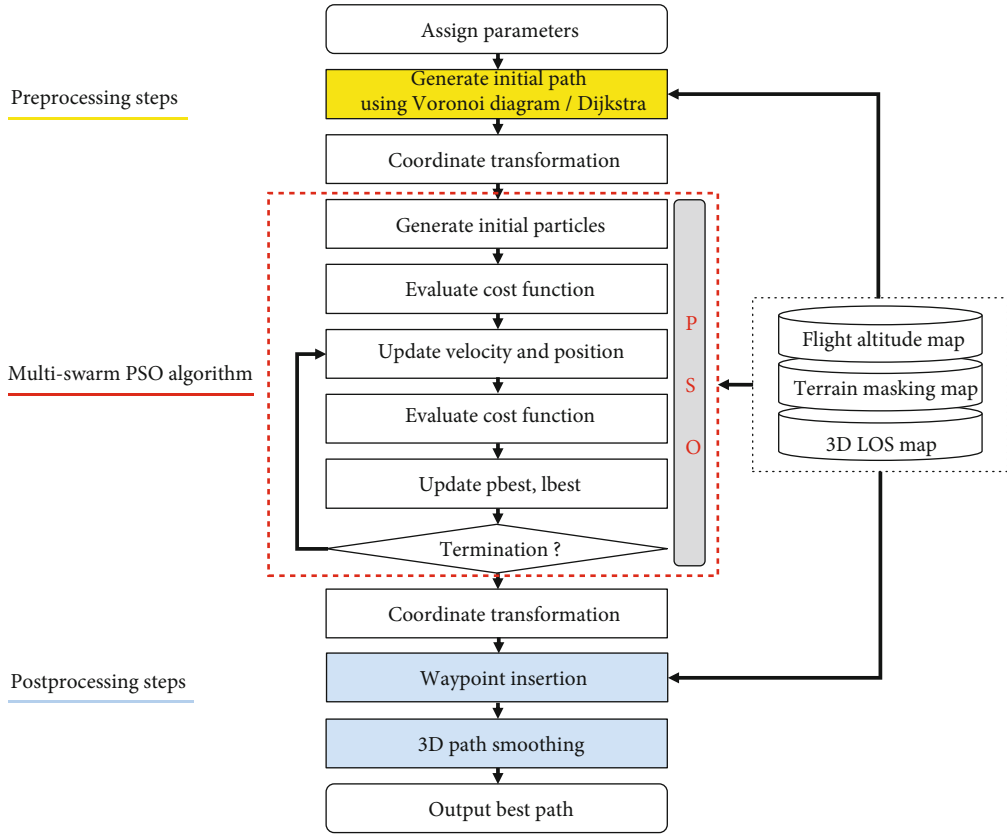


Figure 8: Flowchart of the proposed PSO algorithm.

major differences. In the standard PSO algorithm, a ring topology is used rather than a global topology in the traditional PSO algorithm. In a global topology, every particle shares information from the best particle in the entire swarm. However, in a ring topology, a particle can communicate with only two neighboring particles. This enables the algorithm to escape from a local optimum. The other distinction of the standard PSO algorithm from the traditional PSO algorithm is the introduction of a constriction factor χ. For $\varphi > 4$, the convergence is guaranteed [31]. The following equations are used in the standard PSO algorithm.

$$\begin{cases} \mathbf{v}_i^{k+1} = \chi\left[\mathbf{v}_i^k + c_1\ \mathbf{rand}_1^k .* \left(\mathbf{pbest}_i^k - \mathbf{y}_i^k\right) + c_2\ \mathbf{rand}_2^k .* \left(\mathbf{lbest}_i^k - \mathbf{y}_i^k\right)\right], \\ \chi = \dfrac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \\ \varphi = c_1 + c_2, \end{cases} \tag{10}$$

where **lbest** is a local best vector.

3.2. The Proposed PSO Algorithm. Figure 8 shows the flowchart of the proposed PSO algorithm to find an optimal path for UAV. Whole processes are composed of 3 stages which are preprocessing steps, multi-swarm PSO algorithm, and postprocessing steps. The flight altitude map, the terrain masking map, and the 3D LOS map are prepared in advance. The process starts by assigning the parameters containing a starting point and a destination. As preprocessing steps, the Voronoi diagram and Dijkstra algorithm are conducted to produce the initial path for the PSO algorithm. Then, the coordinate transformation is done to accelerate the searching speed. The multi-swarm PSO algorithm searches for an optimal path with iterations based on transformed coordinates. It differs from the traditional and standard PSO algorithms in that many particles are used and new topology for PSO is suggested. Whole particles are decomposed into multiple swarms which contain sub-swarms. A global topology is used within a sub-swarm, and a ring topology is used between sub-swarms so that fast convergence can occur with a balance between exploration and exploitation. Postprocessing steps

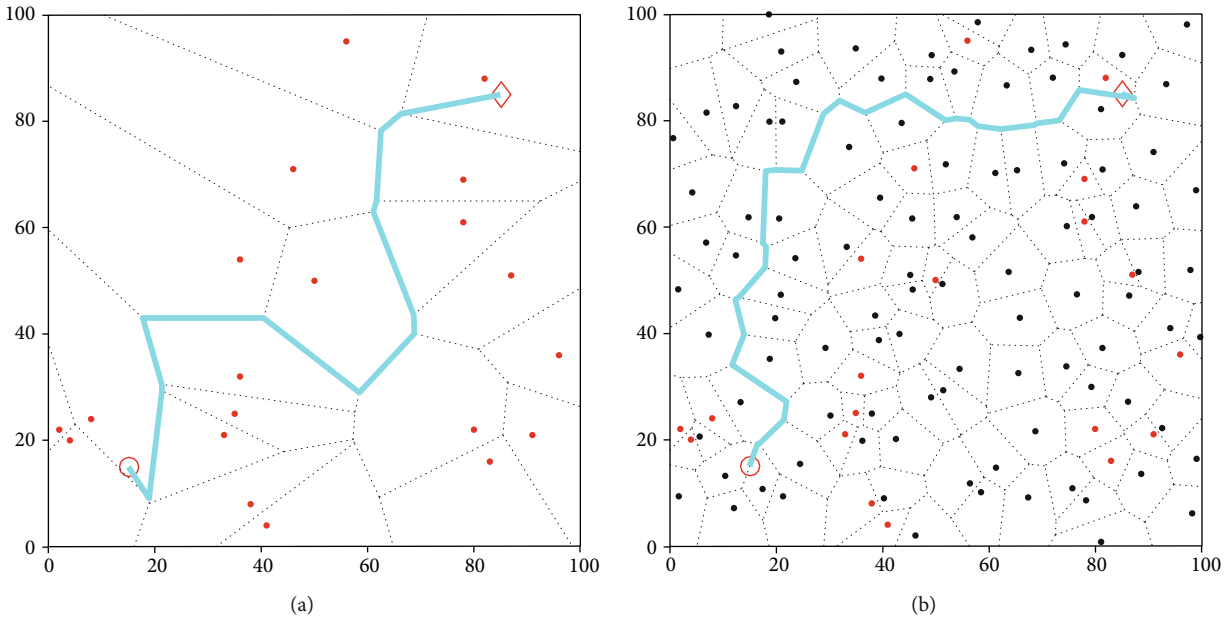


Figure 9: Initial paths generated from the Voronoi diagram and Dijkstra algorithm. (a) Voronoi edges from SAM sites. (b) Voronoi edges from virtual points in addition to SAM sites. A circle denotes a starting point, and a diamond denotes a destination.

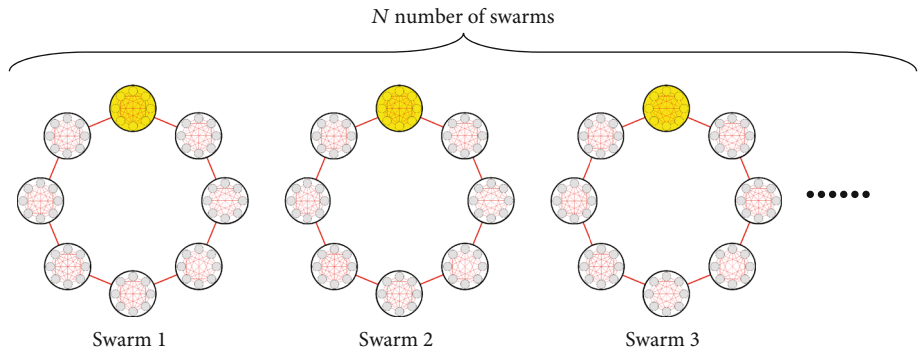


Figure 10: Multi-swarm PSO topologies. Each swarm is divided into sub-swarms. The particles within a sub-swarm have a global topology, and a ring topology is used between sub-swarms.

start by transforming the coordinates back into the original coordinates. Then, the waypoint insertion and the path smoothing are conducted as postprocessing steps.

Preprocessing steps are for generating an initial path which is utilized to initialize the particles in one sub-swarm. The particles initialized from preprocessing steps start to search near the initial path, and the particles of the other sub-swarms start the search in a random fashion. The advantage of the initialization is that the PSO algorithm can have better chances to find an optimal path with less iterations. The Voronoi diagram and Dijkstra algorithm which are simple and fast path planning algorithms are used to produce a path in a fast and reliable way. The optimization criterion is the same as the one which is used for the multi-swarm PSO algorithm. The cost calculation for the Dijkstra algorithm is based on the threat modeling in Section 2. Figure 9 shows two initial paths generated from the Voronoi diagram. In Figure 9(a), only SAM sites are used for obtaining Voronoi edges. The optimal path obtained by using only SAM sites is generated between SAM sites. There may be cases where a path is better away from SAM sites not between SAM sites. Therefore, virtual points in addition to the SAM sites are used to construct Voronoi edges as shown in Figure 9(b). These virtual points are generated by adding more points to make an initial path which can keep a safe distance from SAM sites. The path generated by using virtual points and SAM sites is provided as initial values for the PSO algorithm. The path from the preprocessing steps is adapted to produce an initial path for the PSO algorithm using the 1D interpolation method.

The multi-swarm PSO algorithm is different from other PSO algorithms in several aspects. Many particles are used in the algorithm, and a whole population is composed of multiple swarms. Each swarm has many sub-swarms. A global topology is used within each sub-swarm, and a ring topology is used between the two adjacent sub-swarms. By using the combined topologies, fast convergence is possible with a balance between the exploration and the exploitation.

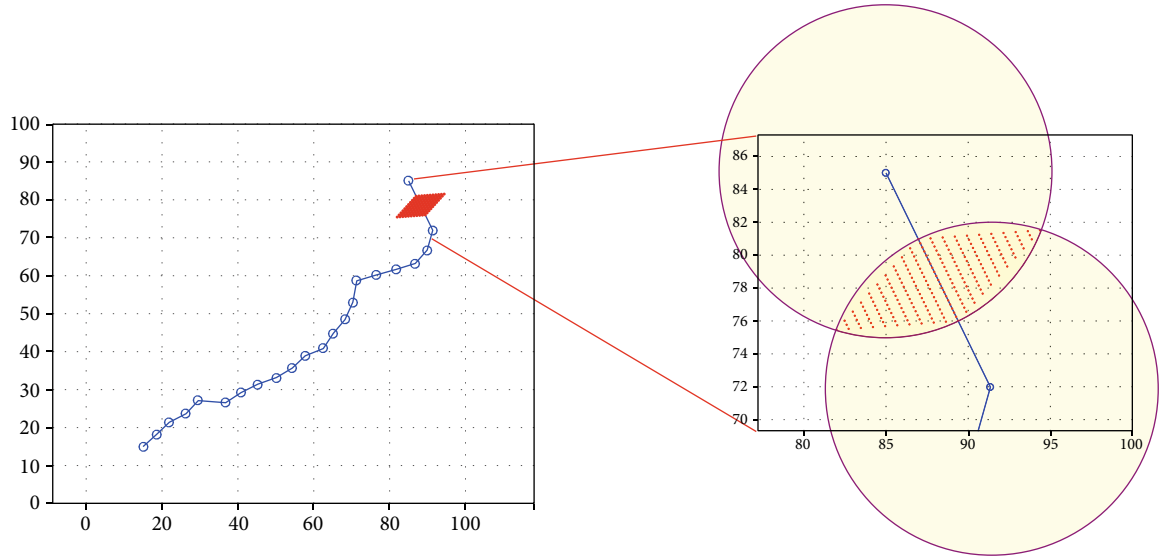
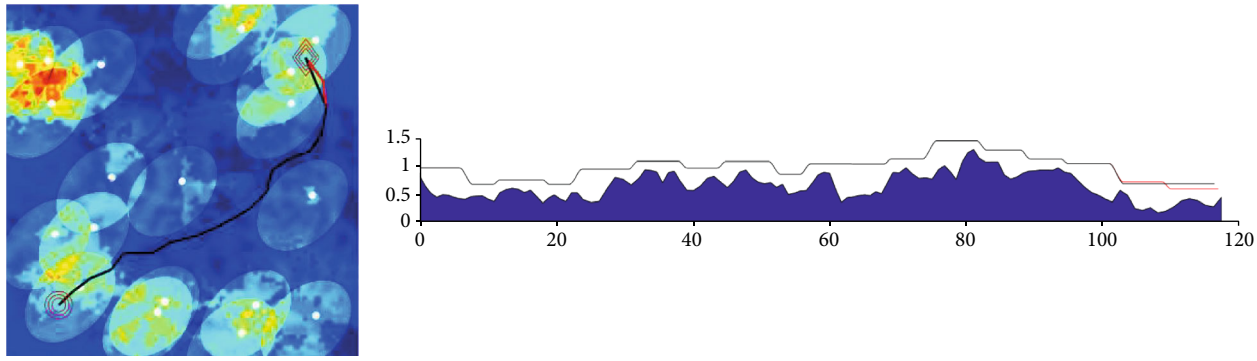


Figure 11: Waypoint insertion for a long segment.



(a) (b)

Figure 12: Original path (black line) and a modified path by waypoint insertion (red line): (a) horizontal path; (b) the altitude according to the travelled distance in the horizontal plane.

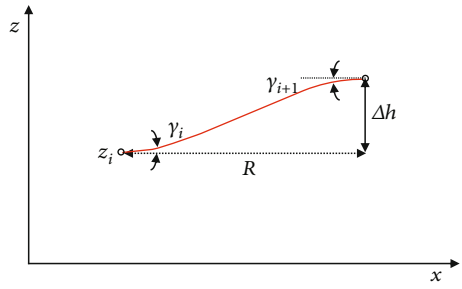


Figure 13: Midcourse guidance trajectory for ascending or descending flight.

There is no data communication between the swarms. Figure 10 shows the topologies used in the multi-swarm PSO algorithm. A large number of particles are exploited for the multi-swarm PSO algorithm. All the particles can be updated simultaneously at each iteration using the vectorization. The number of cost evaluation points should be the same for all segments since all cost tables should be vectorized for parallel computing. The distance between adjacent cost evaluation points for all segments is computed and used for cost calculations. GPU parallel computing technologies can decrease the computation time effectively for massively parallel computations [32].

We have two steps for postprocessing steps. The first step is the waypoint insertion to decrease the vulnerability due to the threats. The other step is 3D path smoothing. Waypoint insertion is implemented in such a way that a long segment is divided into two segments by inserting a new waypoint. If a segment length is larger than two times of the minimum segment length, a waypoint is added. The minimum segment length is a constant value which is determined by the flight characteristics, and the length of every segment along a path should be larger than this value. A new waypoint is determined within the intersected area by two circles whose radius is twice of the minimum segment length. Then, the size of two new segments becomes less than the radius of the circle so that another waypoint insertion will not be necessary for

newly added segments. There are 221 candidate waypoints which are constructed by a rhombus within the intersected area as shown in Figure 11. All the candidate waypoints are evaluated by the same cost functions used in the PSO, and the waypoint with the least cost is chosen for a waypoint insertion. Dynamic constraints are also considered in cost evaluations. Figure 12 shows an example for waypoint insertion. New segments which are indicated as red lines are determined by inserting a waypoint.

The other postprocessing step is 3D path smoothing. The minimum turning radius which can be determined by the flight characteristics is used for the turn. The midcourse guidance law [33] is used for ascending and descending flights. An ascending or descending trajectory is determined according to a downrange distance (R), flight path angles (γ_i , γ_{i+1}), and an altitude change (Δh) as shown in Figure 13.

A trajectory can be described as a third polynomial as a function of a downrange distance as follows.

$$\begin{cases} z = z_i + a_1\left(\dfrac{x}{R}\right) + a_2\left(\dfrac{x}{R}\right)^2 + a_3\left(\dfrac{x}{R}\right)^3, \\ a_1 = \gamma_i R, \\ a_2 = -(2\gamma_i + \gamma_{i+1})R + 3\Delta h, \\ a_3 = (\gamma_i + \gamma_{i+1})R - 2\Delta h. \end{cases} \tag{11}$$

When both flight path angles are zeroes, a trajectory is expressed in a simple form.

$$z = z_i + 3\Delta h\left(\frac{x}{R}\right)^2 - 2\Delta h\left(\frac{x}{R}\right)^3. \tag{12}$$

If small angle assumption and a constant velocity ($dx/dt = V$) are applied to the above equation, an acceleration command (A_c) can be derived. When x is zero or R, a maximum acceleration command ($A_{\max}$) is needed. Finally, a minimum required downrange distance is determined as a function of an altitude difference, a velocity, and a maximum acceleration command.

$$\begin{cases} \dfrac{d^2 z}{dt^2} = A_c = \dfrac{6\Delta h V^2}{R^2}\left(1 - 2\dfrac{x}{R}\right), \\ R_{\min} = \sqrt{\dfrac{6|\Delta h|}{A_{\max}}}V. \end{cases} \tag{13}$$

This minimum downrange distance is required to avoid ground collisions for an ascending or descending trajectory. The minimum downrange distance is determined by the flight characteristics. Figure 14 shows the ascending or descending trajectory for 3D path smoothing. When UAV is ascending, it finishes ascending at the waypoint to avoid the ground collisions. On the other hand, it starts to descend at the waypoint when it is descending. Minimum segment length should be at least two times larger than the required downrange so that a certain segment can contain both ascending and descending trajectories.

Several methods for reducing the computation time are proposed in this paper. In general, the computation time of the PSO algorithm is longer than that of other fast algorithms. To compensate this computation time, the maps are generated beforehand. There is the flight altitude map which is needed to calculate the UAV altitude once horizontal positions are defined. If the SAM sites are defined, the terrain masking map for each radar can be constructed. These two kinds of maps are produced according to horizontal positions on a rectangular grid. The values at a specific position are calculated using the interpolation of the map values. The terrain masking map and the flight altitude map are used for calculating the radar detection cost. The 3D LOS map is generated using the terrain according to the altitude as well as horizontal positions.

Another technique to reduce the computation time is the coordinate transformation which decreases the dimension of PSO. As introduced in Section 2, the coordinate transformation can simplify the computation process by reducing the number of variables used in the PSO algorithm.

4. Simulation Results and Discussions

4.1. Simulation Parameters. The parameters regarding UAV, threats, and the PSO algorithm are described in this section. The level flight with a constant velocity for each segment along the path is assumed so that UAV flies 0.15 km above the maximum altitude of terrain for each segment. As shown in Table 1, the velocity and the minimum turning radius of UAV are assumed to be 0.2 km/sec and 2.35 km, respectively. The minimum segment length is 5 km, which is two times larger than the required downrange distance when the maximum altitude change between the waypoints is assumed to 0.5 km. Sharp turns at the waypoint less than $\pi/2$ radians are not allowed as dynamic constraints of UAV. The penalty cost (P_t) is 10^5 when dynamic constraints are not satisfied. Table 2 shows the starting point and the destination in two sorts of test cases, which are short-range test cases and long-range test cases. Each test case includes 10 different configurations of SAM sites.

Table 3 shows the parameters of SAM. The number of SAM sites is 20, and their locations vary for each test case. The cost is computed based on three parts which are SAM lethal envelope, radar detection, and the 3D LOS map. In this paper, SAM lethal envelope is assumed to have an elliptical shape, which changes the direction of the semimajor axis according to the flight direction of UAV. In addition, it is assumed that the cost $\mathrm{SLE}_{i,j}$ has the value of 100 when UAV is within the envelope. The cost for radar detection is inversely proportional to the 4^{th} power of the distance away from the SAM site. The cost of radar detection for each threat is limited by a value of 100. The maximum value (C_{L}) for the cost of the 3D LOS map is also 100. If we combine all the costs, we can calculate the cost which is used for determining the path.

The simulation parameters for PSO are specified in Table 4. The number of waypoints is 20 or 40 depending on the short-range or long-range test case, which is also the number of particle dimension. The total number of particles

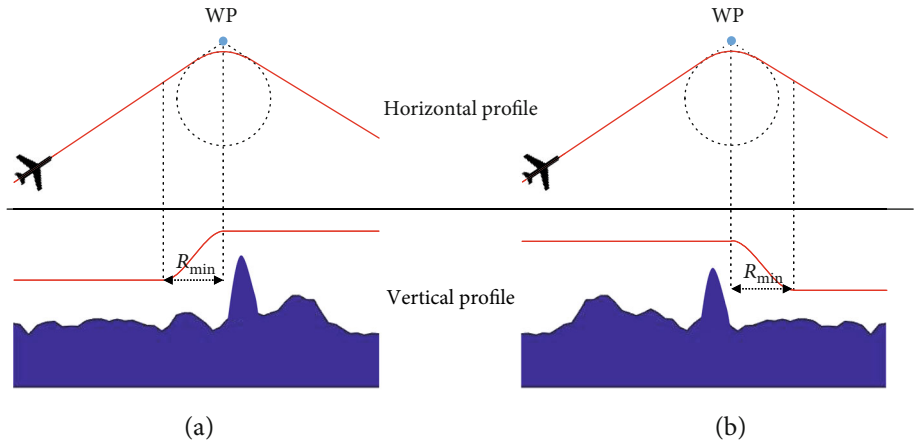


Figure 14: Three-dimensional path smoothing: (a) ascending trajectory; (b) descending trajectory.

Table 1: UAV flight characteristics.

Velocity (V)	Min. turning radius (r_{min})	Min. segment length	Dynamic constraints (T_t)
0.2 km/sec	2.35 km	5 km	$\pi/2$

Table 2: The starting point and the destination for short-range and long-range test cases.

Test case	Starting point	Destination
Short-range	(15, 15)	(85, 85)
Long-range	(15, 15)	(200, 85)

Table 3: Threat model parameters.

Number of SAM sites	SAM lethal envelope (semimajor axis, semiminor axis)	Max. detection range of the SAM radar
20	(15 km, 10 km)	30 km

for a single swarm and the number of iterations are 50 and 1000, respectively, if they are not specified in the simulation results. The number of sub-swarms for a swarm is 5. This means that 10 particles are present for each sub-swarm since the total number of particles is 50. When multiple swarms are used, the total number of particles is the number of swarms multiplied by 50. The constriction factor as well as the individual learning coefficient and the global learning coefficient is given in the table.

4.2. Simulation Results. The proposed PSO algorithm with only one swarm as well as the traditional and standard PSO algorithms is implemented to compare the results. The number of particles is 50. Multi-swarm results are discussed later in this section. An initial path for the proposed PSO algorithm is generated by using the Voronoi diagram and Dijkstra algorithm, and it is used to initialize the particles of one sub-swarm so that other sub-swarms are not affected by the preprocessing steps. The simulations are carried out for 20 different test cases where the locations of SAM sites are chosen differently. Monte Carlo simulations for each PSO method are conducted for 50 times by changing the random numbers for each test case. We analyze the performance of the proposed PSO algorithm as a function of the number of iterations. The performance is calculated from the cost relative to the minimum cost out of the all PSO algorithms for each test case. The minimum cost for each test case is obtained from 50 Monte Carlo simulation results after 1000 iterations. Therefore, the maximum value of the performance is 1.

In Figure 15, the performance results are compared for the deterministic method, the traditional PSO, the standard PSO, and the proposed PSO algorithms. Average performances are represented for short-range test cases and long-range test cases. The proposed PSO algorithm shows fast convergence and better results with exploration and exploitation balance. The preprocessing steps are effective particularly for test cases where it is difficult to overcome a local optimum. The postprocessing steps are efficient for a path with long segments. The postprocessing steps affect a path for 8 test cases of short-range and 10 test cases of long-range. We do not need many iterations to reach a satisfactory performance using the proposed PSO algorithm, and the computation time can be also saved. At 200 iterations, the average performance of the proposed PSO algorithm is 0.93 and 0.92 for short-range test cases and long-range test cases, respectively. On the contrary, the traditional PSO algorithm converges prematurely and does not get better with more iterations. The standard PSO results show that the performance gets better with more iterations while the performance is poor with less iterations. The traditional and standard PSO algorithms suffer more on the long-range test cases with 40 variables. Therefore, the results indicate that the proposed algorithm shows much better performance than the other algorithms even though the iterations of the proposed method are much less.

Figure 16 shows the effect of preprocessing steps. All 3 PSO algorithms are initialized by a path from preprocessing steps. A single swarm with 50 particles is used. The difference is mainly from the topologies used in the PSO algorithms.

Table 4: PSO parameters.

Particle dimension (D)	Number of particles	Number of subswarms	Number of iterations	Constriction factor (χ)	Individual learning coeff. (c_1)	Global learning coeff. (c_2)
20 or 40	50	5	1000	0.72984	2.05	2.05

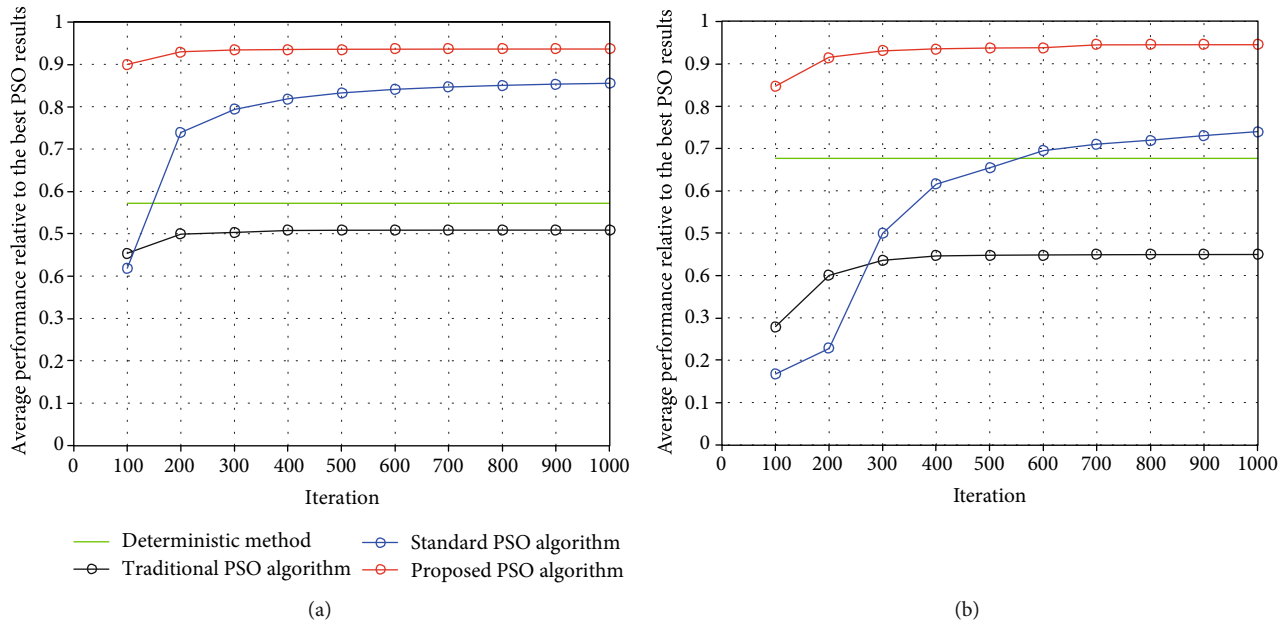


Figure 15: Average performance comparison according to the number of iterations for the short-range test cases (a) and the long-range test cases (b).

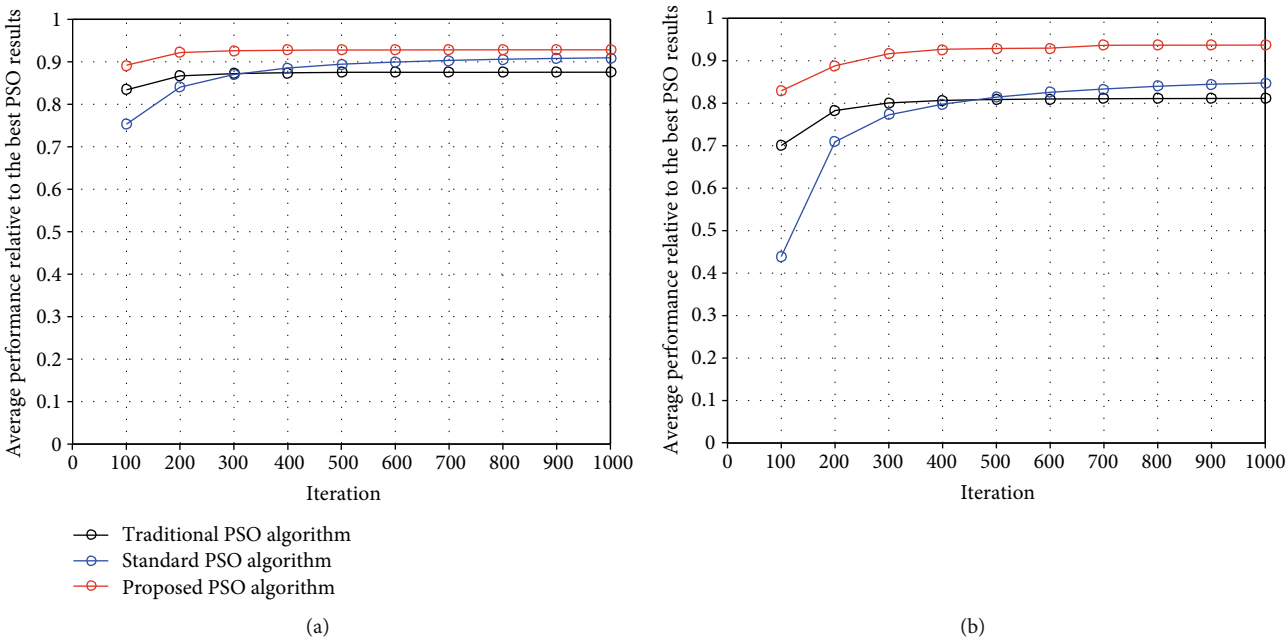


Figure 16: Average performance comparison for preprocessing steps according to the number of iterations for the short-range test cases (a) and the long-range test cases (b). Preprocessing steps are applied, and postprocessing steps are not applied for all 3 PSO algorithms.

Table 5: Mean costs and standard deviations for 20 test cases with 50 Monte Carlo simulations.

Test case		Proposed PSO		Standard PSO		Traditional PSO		Voronoi+Dijkstra
		Mean	Std.	Mean	Std.	Mean	Std.	
Short-range	1	10866	327	14235	2267	22219	23504 (4)	20378
	2	11350	216	16119	1915	23805	25718 (5)	17307
	3	10749	456	13181	1150	19171	20889 (3)	21756
	4	15157	769	18563	1666	26258	24972 (5)	21965
	5	10537	401	11768	646	13784	12641 (1)	17178
	6	12608	360	17257	3308	30082	26381 (6)	18965
	7	14386	683	21110	2723	26822	19301 (3)	22642
	8	11195	598	13923	1306	20808	23666 (4)	17075
	9	13491	540	14343	717	21262	23528 (4)	20666
	10	10738	165	13077	1528	26392	30103 (7)	19016
Long-range	1	18778	603	91193	22125 (43)	57775	34965 (20)	25905
	2	22083	16079 (2)	65456	34974 (25)	51606	36769 (18)	26516
	3	18075	720	80254	32012 (36)	33759	24915 (6)	26599
	4	20048	444	77579	31665 (33)	43302	28814 (10)	26974
	5	16785	252	66009	37244 (27)	32543	27653 (7)	23331
	6	19348	326	79772	31287 (35)	45494	31164 (12)	24194
	7	17464	425	85126	28380 (39)	50753	35890 (17)	24646
	8	15881	341	54988	37221 (20)	36265	30304 (9)	21285
	9	17307	328	65143	36748 (26)	38562	31154 (10)	25634
	10	17629	620	80749	31246 (36)	39058	29049 (9)	22688

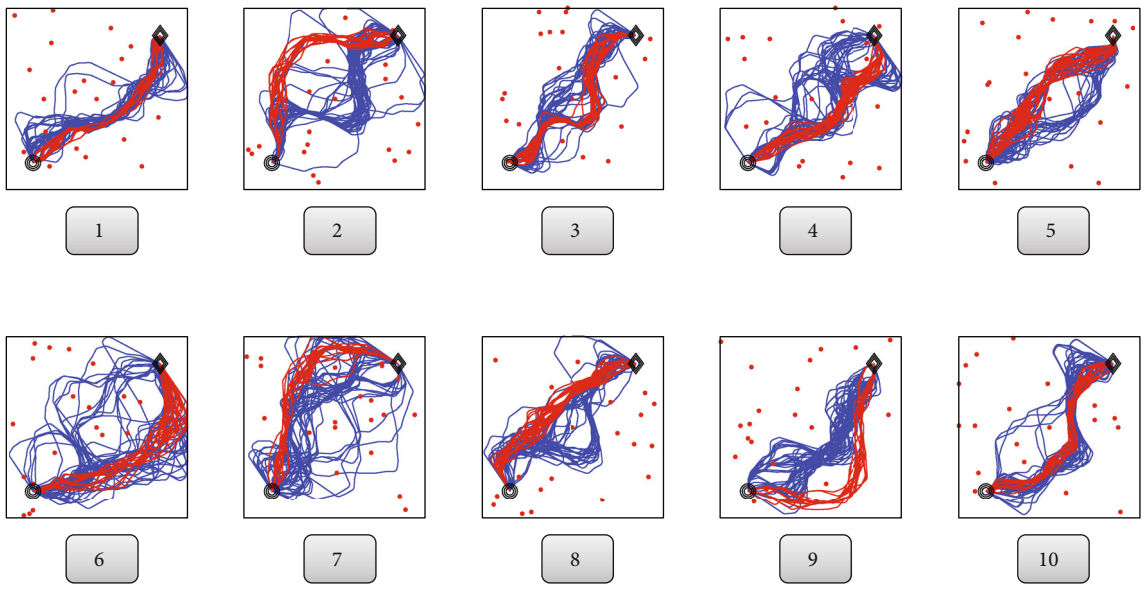


Figure 17: The paths of 50 Monte Carlo simulations at 200 iterations for short-range test cases. The blue lines and red lines represent the standard PSO results and the proposed PSO results, respectively.

The performance is improved by the initialization from the Dijkstra algorithm for all PSO algorithms. The proposed algorithm shows better results for both short-range and long-range test cases than the other algorithms in terms of the average performance. As we can see in the figure, the traditional PSO algorithm with global topology shows the fastest convergence. Premature convergence can occur so that the performance is not good enough. On the other hand, the standard PSO algorithm with a ring topology shows the slowest convergence. The performance gets better with more iterations. The proposed PSO algorithm shows the convergence in between two other algorithms.

Table 5 compares the statistical results at 200 iterations from the proposed PSO algorithm, the standard PSO

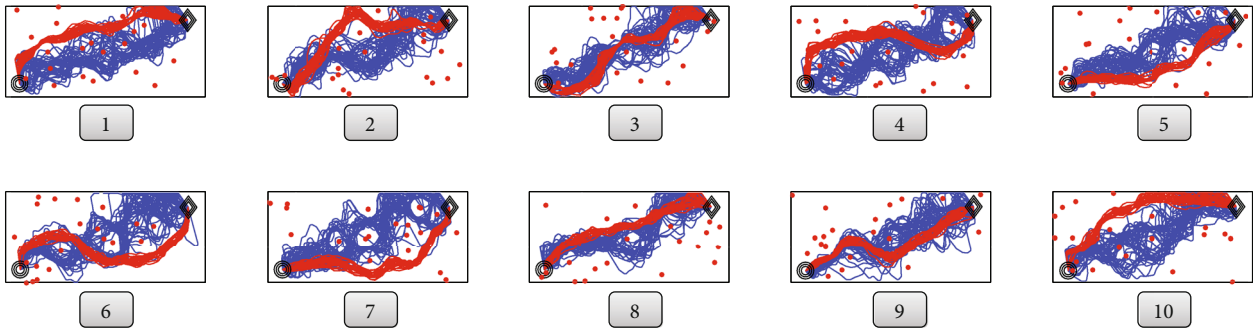


Figure 18: The paths of 50 Monte Carlo simulations at 200 iterations for long-range test cases.

algorithm, the traditional PSO algorithm, and the deterministic method which uses the Voronoi diagram and Dijkstra algorithm. The proposed PSO algorithm shows better results than the other two PSO algorithms and the deterministic method for all the test cases in terms of the mean cost and the standard deviation. In particular, we see that the standard deviations of the proposed method are much smaller than those of the other two methods for the long-range test cases where it is more difficult to find solutions. Statistical difference between the proposed PSO algorithm and the standard PSO algorithm is tested using independent t-tests. The t-test results show that the proposed method is statistically different from the standard PSO for all the test cases since p values are significantly less than 0.05. The number of Monte Carlo simulations in which dynamic constraints are not satisfied is shown in parenthesis next to the standard deviations. Dynamic constraints are not satisfied at least one time for all the cases of the traditional PSO algorithm. The standard PSO results are worse for the long-range test cases. The dynamic constraints are not satisfied for test case #2 of long-range even though the proposed PSO algorithm is used.

We plot 50 Monte Carlo simulation paths for each test case to show the consistency of the results. In Figure 17, the best paths for each test case are plotted for the proposed PSO algorithm and the standard PSO algorithm at 200 iterations. The standard PSO results (blue lines) show quite different paths from each other. However, the proposed PSO results (red lines) show very similar paths which are guided by the preprocessing steps. In Figure 18, the effectiveness of the preprocessing steps is more significant for long-range test cases where it is more difficult to find an optimal path with more variables.

Even though the results are quite promising for the proposed PSO algorithm using only a single swarm with 50 particles, there could be some cases which do not satisfy the dynamic constraints. In addition, there are differences in results due to the random numbers which are used for PSO algorithm itself when we can see the Monte Carlo simulation results. The multi-swarm PSO algorithm can ease these problems and increase the performance better. Table 6 shows the relations between the number of swarms and the total number of particles used for the multi-swarm PSO algorithm. A swarm consists of 50 particles, and the total number of particles is 2000 for 40 swarms. Figure 19 shows the performance results according to the number of swarms used in the simulations at 200 iterations. The results of the deterministic method and the traditional PSO and the standard PSO algorithms are compared with those of the proposed PSO algorithm. For both short-range and long-range test cases, we see the better results with more swarms for all PSO algorithms. The results are improved even more for the traditional PSO and the standard PSO algorithms with more swarms. The proposed PSO algorithm can also improve the performance and reduce the possibility of the cases which do not satisfy the dynamic constraints. The average performance of the proposed PSO algorithm with 10 swarms is 0.97 and 0.96 for short-range test cases and long-range test cases, respectively. In addition, the variance of the performance due to the randomness in the PSO algorithm can be reduced significantly. The increase of the computation time as a disadvantage of using multiple swarms can be moderated using the vectorization and the parallel computing.

Table 6: The relations between the number of swarms and the total number of particles used for the multi-swarm PSO algorithm.

The number of swarms	1	10	20	30	40
The number of particles	50	500	1000	1500	2000

The cost is analyzed in terms of the number of cost evaluation points between two adjacent waypoints. The computation time is increased with more cost evaluation points. Less cost evaluation points may not consider the effects of the SAM sites properly. Figure 20(a) shows the boxplots of the cost ratio in percentage according to the number of cost evaluation points for 2000 different paths. The cost ratio is defined as the total cost for a certain number of cost evaluation points divided by the total cost for 100 cost evaluation points. The cost ratio is set as 100% when the number of cost evaluation points is 100. The average value of the cost ratio approaches 100% for larger cost evaluation points. The average cost ratio is 97% for 10 cost evaluation points, which is pretty similar to the result of 100 cost evaluation points. Figure 20(b) shows that the computation time ratio is 1.17 and 3.37 for 10 and 100 cost evaluation points when the elapsed time ratio is 1 for 2 cost evaluation points. Therefore, optimal cost evaluation points can be determined in terms of the cost ratio and the elapsed time ratio.

The path considering dynamic SAM lethal envelope which changes the direction of the semimajor axis according

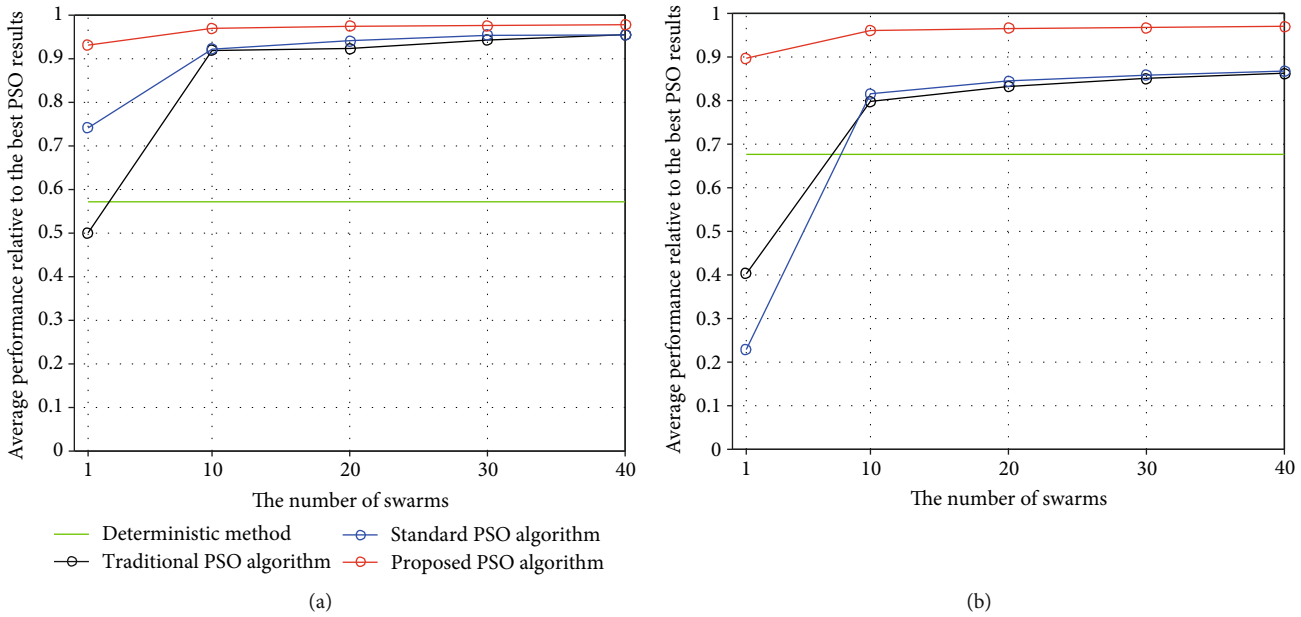


Figure 19: Average performance comparison according to the number of swarms used in the multi-swarm PSO algorithm for the short-range test cases (a) and the long-range test cases (b).

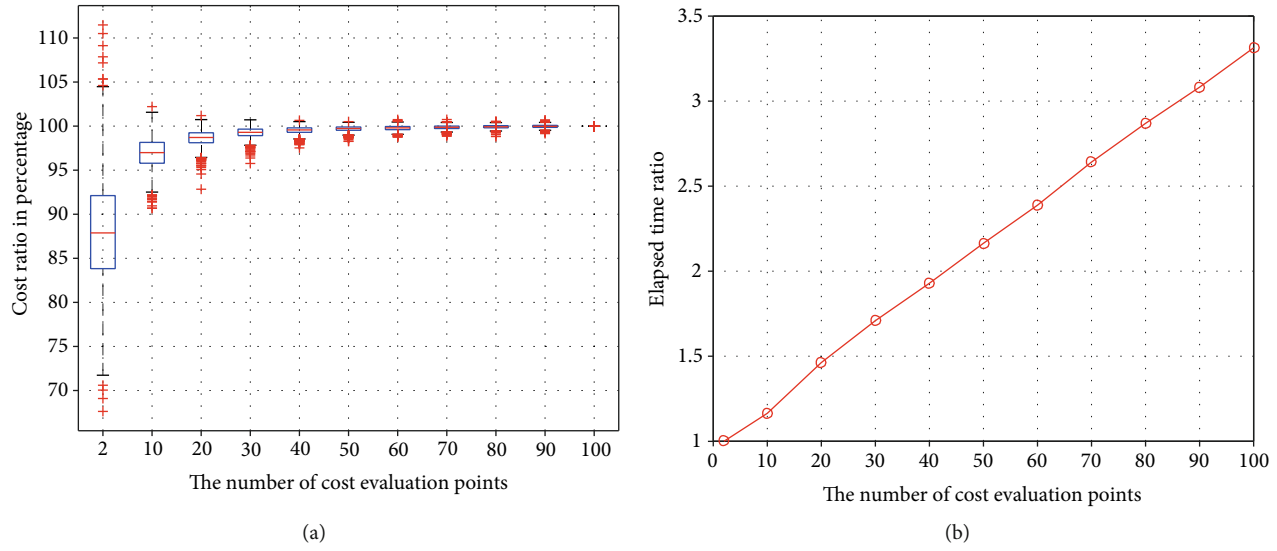


Figure 20: The cost ratio (a) and the elapsed time ratio (b) according to the number of cost evaluation points between two adjacent waypoints.

to the flight direction of UAV is different from that of static SAM lethal envelope. Figure 21 shows the path comparison between the dynamic and static SAM lethal envelopes. Static SAM lethal envelope is assumed to be parallel to the line connecting the starting point and the destination. Although the UAV position of A or B along the path for dynamic SAM lethal envelopes is within static SAM lethal envelope, UAV is actually not located within dynamic SAM lethal envelope as indicated in red dots.

The elliptical shape of SAM lethal envelopes is assumed in the simulations. How can we implement SAM lethal envelope if it has an arbitrary shape rather than an ellipse? If SAM lethal envelope cannot be defined using the equation due to a complex shape, the maps describing SAM lethal envelope can be used instead of the equation. The maps should be generated for various headings of UAV since SAM lethal envelope changes its direction according to the flight direction of UAV. The computation time for path planning can be reduced when the maps are prepared in advance.

5. Conclusions

The method to find an optimal flight path for UAV in adversarial environments is presented in this paper. Hostile environments include SAM lethal envelope, radar detection, and unknown threats. Dynamic SAM lethal envelope is

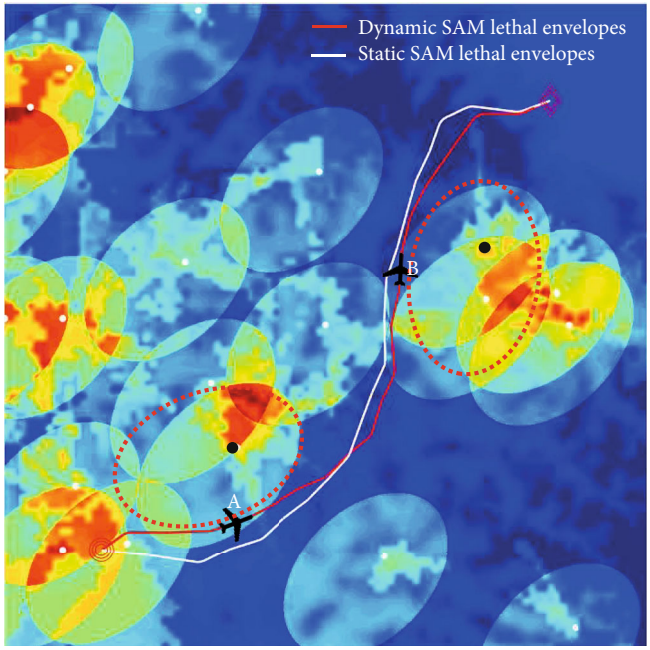


Figure 21: Path comparison between dynamic and static SAM lethal envelopes.

assumed to have an elliptical shape and changes its direction according to the heading of UAV. Terrain masking, RCS, and dynamic constraints of UAV are taken into account to calculate the cost for path planning. The improved PSO algorithm is suggested by using preprocessing steps, multi-swarm PSO algorithm, and postprocessing steps. Preprocessing steps use the Voronoi diagram and Dijkstra algorithm to generate an initial path which is informed to the particles in one subswarm. The multi-swarm PSO algorithm differs from other PSO algorithms in that many particles are decomposed into multiple swarms for parallel computing. Each swarm has sub-swarms so that a good balance between exploration and exploitation can be achieved. In postprocessing steps, a new waypoint is inserted for a long segment to decrease the vulnerability even more. Finally, 3D waypoint smoothing is carried out by using the midcourse guidance law. The computation time for the proposed PSO algorithm is reduced by using the map generation, the coordinate transformation, and the GPU implementation.

The results show that the suggested method is better than existing PSO algorithms in terms of the mean cost and the standard deviation. The average performance of the proposed PSO algorithm with a single swarm at 200 iterations is 0.93 and 0.92 for short-range test cases and long-range test cases, respectively. The performances with 10 swarms become 0.97 and 0.96 while satisfying all the dynamic constraints. The path generated by using dynamic SAM lethal envelope which is a more realistic model is different from that of static SAM lethal envelope.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] H. Shakhatreh, A. H. Sawalmeh, A. al-Fuqaha et al., "Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[2] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of Unmanned Aerial Vehicles*, Springer, 2015.

[3] Y. Liu, X. Zhang, X. Guan, and D. Delahaye, "Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization," *Aerospace Science and Technology*, vol. 58, pp. 92–102, 2016.

[4] S. Butenko, R. Murphey, and P. M. Pardalos, *Cooperative Control: Models, Applications and Algorithms*, Springer Science & Business Media, 2013.

[5] W.-Y. Shin, J.-J. Shin, B.-J. Kim, and K.-R. Jeong, "Line segment selection method for fast path planning," *International Journal of Control, Automation and Systems*, vol. 15, no. 3, pp. 1322–1331, 2017.

[6] M. Zhang, C. Su, Y. Liu, M. Hu, and Y. Zhu, "Unmanned aerial vehicle route planning in the presence of a threat environment based on a virtual globe platform," *ISPRS International Journal of Geo-Information*, vol. 5, no. 10, p. 184, 2016.

[7] Changwen Zheng, Lei Li, Fanjiang Xu, Fuchun Sun, and Mingyue Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 609–620, 2005.

[8] S. A. Bortoff, "Path planning for UAVs," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 1no. 6, pp. 364–368, Chicago, IL, USA, USA, June 2000.

[9] P. T. Kabamba, S. M. Meerkov, and F. H. Zeitz, "Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 2, pp. 279–288, 2006.

[10] F. W. Moore, "Radar cross-section reduction via route planning and intelligent control," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 5, pp. 696–700, 2002.

[11] S.-H. A. Woo, J.-J. Shin, and J. Kim, "Implementation and analysis of pattern propagation factor based radar model for path planning," *Journal of Intelligent & Robotic Systems*, vol. 96, no. 3-4, pp. 517–528, 2019.

[12] S. Lim and H. Bang, "Waypoint planning algorithm using cost functions for surveillance," *Pharmacopoeia Londinensis*, vol. 11, no. 2, pp. 136–144, 2010.

[13] S.-S. Jan and Y.-H. Lin, "Integrated flight path planning system and flight control system for unmanned helicopters," *Sensors*, vol. 11, no. 8, pp. 7502–7529, 2011.

[14] L. De Filippis, G. Guglieri, and F. Quagliotti, "A minimum risk approach for path planning of UAVs," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 203–219, 2011.

[15] L. De Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for UAVS in 3D environments," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 247–264, 2012.

[16] P. Chandler, S. Rasmussen, and M. Pachter, "UAV cooperative path planning," in *AIAA Guidance, Navigation, and Control*

Conference and Exhibit, p. 4370, Dever, CO, U.S.A, August 2000.

[17] K. Judd and T. McLain, "Spline based path planning for unmanned air vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 4238, Montreal,Canada, August 2001.

[18] T. Turker, O. K. Sahingoz, and G. Yilmaz, "2D path planning for UAVs in radar threatening environment using simulated annealing algorithm," in *2015 International Conference on Unmanned Aircraft Systems*, pp. 56–61, Denver, CO, USA, June 2015.

[19] Y. Eun and H. Bang, "Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm," *Journal of Aircraft*, vol. 46, no. 1, pp. 338–343, 2009.

[20] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[21] N. Özalp and O. K. Sahingoz, "Optimal UAV path planning in a 3D threat environment by using parallel evolutionary algorithms," in *2013 International conference on unmanned aircraft systems (ICUAS)*,, pp. 308–317, Atlanta, GA, USA, May 2013.

[22] M. Bagherian and A. Alos, "3D UAV trajectory planning using evolutionary algorithms: a comparison study," *The Aeronautical Journal*, vol. 119, no. 1220, pp. 1271–1285, 2015.

[23] Y. Qu, Y. Zhang, and Y. Zhang, "A global path planning algorithm for fixed-wing UAVs," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 3-4, pp. 691–707, 2018.

[24] L. Wang, X. Zhang, P. Deng, J. Kang, Z. Gao, and L. Liu, "An energy-balanced path planning algorithm for multiple ferrying UAVs based on GA," *International Journal of Aerospace Engineering*, vol. 2020, 15 pages, 2020.

[25] Z. Cheng, E. Wang, Y. Tang, and Y. Wang, "Real-time path planning strategy for UAV based on improved particle swarm optimization," *JCP*, vol. 9, no. 1, pp. 209–214, 2014.

[26] Y. Liu, X. Zhang, X. Guan, and D. Delahaye, "Potential odor intensity grid based UAV path planning algorithm with particle swarm optimization approach," *Mathematical Problems in Engineering*, vol. 2016, Article ID 7802798, 2016.

[27] C. Xu, H. Duan, and F. Liu, "Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning," *Aerospace Science and Technology*, vol. 14, no. 8, pp. 535–541, 2010.

[28] B. R. Mahafza, *Radar Systems Analysis and Design Using MATLAB Second Edition*, Chapman and Hall/CRC, 2005.

[29] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2011.

[30] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[31] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007*, pp. 120–127, Honolulu, HI, USA, April 2007.

[32] Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," in *2009 IEEE Congress on Evolutionary Computation*, pp. 1493–1500, Trondheim, Norway, May 2009.

[33] Y.-I. Lee, S.-H. Kim, and M.-J. Tahk, "Optimality of linear time-varying guidance for impact angle control," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2802–2817, 2012.