UAVs Path Planning by Particle Swarm Optimization Based on Visual-SLAM Algorithm

Umair Ahmad Mughal, Ishtiaq Ahmad, Chaitali J. Pawase, and KyungHi Chang

Abstract Intelligent 3-D path planning is a crucial aspect of an unmanned aerial vehicle's (UAVs) autonomous flight system. In this chapter, we propose a two-step centralized system for developing a 3-D path-planning for a swarm of UAVs. We trace the UAV position while simultaneously constructing an incremental and progressive map of the environment using visual simultaneous localization and mapping (V-SLAM) method. We introduce a corner-edge points matching mechanism for stabilizing the V-SLAM system in the least extracted map points. In this instance, a single UAV performs the function using monocular vision for mapping an area of interest. We use the particle swarm optimization (PSO) algorithm to optimize paths for multi-UAVs. We also propose a path updating mechanism based on region sensitivity (RS) to avoid sensitive areas if any hazardous events are detected during the execution of the final path. Moreover, the dynamic fitness function (DFF) is developed to evaluate path planning performance while considering various optimization parameters such as flight risk estimation, energy consumption, and operation completion time. This system achieves high fitness value and safely arrives at the destination while avoiding collisions and restricted areas, which validates the efficiency of proposed PSO-VSLAM system as demonstrated by simulation results.

Keywords Visual-SLAM · PSO · Path planning · Autonomous aerial vehicles · UAV

1 Introduction

The ability of an autonomous aerial vehicle to navigate in an unknown environment while simultaneously building a progressive map and localizing itself is a prominent research topic in robotics. Because of the practical uses of simultaneous localization and mapping (SLAM), research has been conducted [1]. Advances in vision-based

U. A. Mughal · I. Ahmad · C. J. Pawase · K. Chang (✉)
Department of Electrical and Computer Engineering, INHA University, Incheon 22212, South Korea
e-mail: khchang@inha.ac.kr

Z. Kaleem et al. (eds.), *Intelligent Unmanned Air Vehicles Communications for Public Safety Networks*, Unmanned System Technologies,
https://doi.org/10.1007/978-981-19-1292-4_8

SLAM algorithms assess the robot's position and generate the terrain as the robot of interest moves [2]. Many SLAM systems in the literature include a diverse set of sensors, including Laser Range Finders (LRF), inertial measurement units (IMU), GNSS receivers, magnetometers, optical flow sensors (OFS), barometers, and Light Detection and Ranging (LiDAR) [3, 4]. Single camera SLAM systems, on the other hand, have gained in popularity in recent years due to their light weight, low cost, and variety of applications in complex environments [5, 6]. In this regard, monocular visual-SLAM has gotten attention for UAV applications since it provides fully autonomous systems in a range of challenging settings without the usage of external positioning systems. UAVs are commonly used for traffic monitoring, health services, search and rescue, security, and surveillance [7–9]. UAVs enhance wireless network coverage, capacity, and efficiency by serving as base stations [10].

Path planning algorithms are designed to find the optimum path based on a set of constraints and objectives (such as terrain constraints and collision avoidance, energy consumption, flight risk, etc.). As a result, path planning must take into account not only limitations and objectives, but also the possibility of dangerous events that occurred unexpectedly along the UAV's path. We propose the region sensitivity (RS) to reduce unconditional hazards by allowing the UAV to recognize an unsafe region and optimize its path to the destination. The focus of this research is to provide a framework for determining the best path to take using monocular vision maps. A visual-SLAM (VSLAM) approach builds an incremental map of the environment while continuously tracking the camera's position. Following that, the resulting map is analyzed and used as input for an optimization algorithm.

The PSO framework is easier to implement and requires less time to compute than other metaheuristic search algorithms. It is also better at handling nonlinear challenges than other heuristic algorithms like ant colony optimization (ACO), Genetic algorithm (GA), and an evolutionary technique (EA). Because the GA is fundamentally discrete, i.e., it encodes to design discrete variables, it has a high computing cost, whereas the PSO is inherently continuous and can be easily modified to handle discrete design variables. As a result, we utilize PSO since it converges efficiently in a dynamic environment. The particle is treated as an integrated individual in the PSO framework, representing a candidate solution. As a result, the performance of all particles defines the global best particle. To analyze a feasible path, the PSO planner evaluates the quality of the entire path rather than a single waypoint.

1.1 Main Contributions

This chapter aims to develop a system that generates the best paths for multiple UAVs to safely arrive at their destinations, even when GPS is unavailable. To build an incremental and progressive map of the surrounding environment, we designed a two-step centralized system based on visual-SLAM. The constructed terrain map in the form of a points cloud is loaded into the proposed multiple-path UAVs optimization planner. To stabilize the system in the least textured environment, we use

the Canny and Harris detectors at the same time. We proposed a dynamic fitness function (DFF) as a joint cost determinant, which contains multiple optimization indexes, such as flight risk estimation, energy consumption, operation completion time, and numerous constraints, such as UAV constraints, which consider the physical limitations of the UAVs, and environmental constraints, which also consider the surrounding conditions. To address unexpected hazardous events, we've presented a path-updating system based on the RS, which allows the UAV to identify an unstable location and optimize its path accordingly. Based on the RS and DFF, the proposed optimization planner utilizes the PSO to compute the fitness of each path. All of these factors contribute to the practicality of our proposed methodology for path planning of multiple UAVs.

1.2 Related Work

SLAM and PSO technologies are often used in research involving underwater, interior, and outdoor environments. The authors of [11] utilize active SLAM for deep reinforcement learning-based robot path planning. The convolutional residual network is used to detect obstacles in the path. The suggested approach employs the Dueling DQN algorithm for obstacle avoidance while also employing the FastSLAM technique to create a 2D map of the surrounding area. Similarly, the authors of [12] use stereo vision-based active SLAM to locate, navigate, and map their environment. To avoid obstacles and complete the task effectively, the cognitive-based adaptive optimization algorithm is introduced. The main focus of the approaches in [11, 12] is on the complete robot task while detecting and avoiding the environment's obstacles.

In [13], the author recommends using a visual-SLAM technique to build an incremental map of the terrain for surveillance. For path planning, the author offers the Cognitive-based Adaptive Optimization (CAO) algorithm. A monocular-inertial SLAM is proposed in [14]. To augment the monocular camera's sensing cues with inertial measurement unit (IMU). PSO method was used in a hazard exploration scenario for a network of UAVs in [15]. The new and improved PSO is proposed as dynamic PSO for UAV networks (dPSO-U). UAVs use delay tolerant networking (DTN) for sharing information. The solution simply evaluates the optimum UAV combinations to thoroughly explore the environment. The 5G network is enhanced with multiple UAVs in [16]. The UAVs serve as a link between the users and the cellular base station. The designed approach's major goal is to position the UAVs in the best possible position to maximize the communication coverage ratio. The authors offer per-drone iterated PSO (DI-PSO) system that utilizes PSO to find the optimum position for each drone. In our method, the UAVs function as individual PSO particle. A group of unmanned aerial vehicles (UAVs) tackles a forest fire in [17]. Before the mission begins, the target locations are assumed to be known. Using an auction-based algorithm, the UAVs were assigned to the various fire areas. The UAVs then employ the centralized PSO algorithm, as well as the parametrization and

time discretization (CPTD) algorithm, to compute the best paths to the designated fire sites.

In [18], an improved PSO algorithm is used for real-time path planning of a single UAV. Work falls under the low-level category of trajectory planning because it involves avoiding moving obstacles. The NBVP [19] is relevant to this paper. Within the planning loop, it employs the RRT technique. A tree node is used to retrieve visual data from the depth sensor. During planning, a small fraction of the best view is executed in each iteration, enabling the trajectory to be adapted to the plan between iterations as a new explored map.

Our previous work [20] examined the environmental and physical characteristics of the surroundings. However, we present a dynamic fitness function (DFF), which involves various optimization factors to handle environmental constraints including terrain limitations, restricted areas, collision avoidance, etc. Moreover, we propose RS to tackle any unexpected hazardous event during UAV flight. To find the optimal DFF and RS system designs, we employ a monocular vision-based SLAM technique. In [21] authors developed an enhanced PSO (IPSO) for robot path planning. The authors evaluate three alternatives in two different environments: PSO, artificial potential field (APF), and IPSO. In [22], the authors developed the adaptive selection mutation limited differential evolution method for path planning in disaster environments. A single objective evolutionary technique, based on reference points, is presented in [23]. The author also developed a hybrid grey wolf optimization technique for UAV path planning in [24]. In the literature, different system parameters were generated from various system philosophies and objectives [25].

Challenges of 3-D UAV placement, such as resource and power allocation, trajectory optimization, and user association are discussed in [26]. This challenge becomes considerably more complicated as the height of the UAV changes, changing the channel conditions and reducing coverage due to severe co-channel interference. The authors proposed optimizing the 3-D UAV placement and path-loss compensation factor for various UAV deployment heights in the suburban setting in order to provide a solution. The authors of [27] suggested a rapid K-means-based user clustering model and jointly optimum power and time transfer-ring allocation that can be used in the real system by deploying UAVs as flying base stations for real-time network recovery and maintenance during and after disasters. Nguyen et al. [28] presented a unique approach based on deep reinforcement learning for finding the best solution for energy-harvesting time scheduling in UAV-assisted D2D communications. The article [29] investigated wireless systems by using a full-duplex (FD) unmanned aerial vehicle (UAV) relay to allow two adjacent base stations to communicate with users and are far distant. In order to increase user performance, non-orthogonal multiple access (NOMA) aided networks and multiple-antenna user design are also investigated. For delay-sensitive communication in UAVs, a disaster resilient three-layered architecture for PS-LTE (DR-PSLTE) is presented in [30].

2 Visual-SLAM Framework

Most vision-based SLAM systems employ a corner features detector, such as the Harris corner detector. In a non-textured scene, the corner detector cannot identify enough feature points. As a solution, we present the corner-edge point system, which employs edge points as well. The edge-point is the detected point on the edge segment. Our method recognizes corner and edge points by comparing the 3D points of the next image and estimating the camera's position by comparing the 3D points of the next image. In this method, the camera's trajectory and a 3D map are produced. In addition to robustness, it provides a detailed representation of the object, which improves the modeling process of surface detection and reconstruction.

2.1 Approach

Correspondence between the points may lead to multiple matches, including outliers. Random sampling consensus (RANSAC) [31] handles inliers, outliers, dividing data using perspective projection [32]. The large matching errors are eliminated by the progressive sample consensus PROSAC algorithm [33]. In the beginning, we estimate the trajectory of the camera with small detected points, and afterwards, we use a coarse-to-fine approach to refine the trajectory and feature point correspondence by progressively increasing the points. The overall approach to constructing a map using the visual- SLAM system can be seen in Fig. 1.

2.1.1 Keypoint Matching

Most computer vision applications require Structure from Motion (SfM), Multi-view Stereo (MvS), image registration, and image retrieval. The technique begins with keypoint detection and description and then proceeds on to keypoint matching. A descriptor is a multidimensional vector that denotes the keypoints in space. As a result, the keypoint is identified, which is then projected on the images from two different perspectives. First, we apply acceleration segment characteristics to find keypoints (FAST). The edge locations are then determined using the well-known Harris Corner detector [34] and Canny edge detector [35]. To eliminate outliers, the robust independent elementary features (BRIEF) descriptor is oriented around the gradient. Due to their lower processing complexity and higher accuracy compared to other detectors and descriptors [36]. The SIFT has the lowest matching rate of 31.8% in 0.25 s, while the ORB combines FAST and BRIEF to have the highest matching rate of 49.5% in 0.02 s.

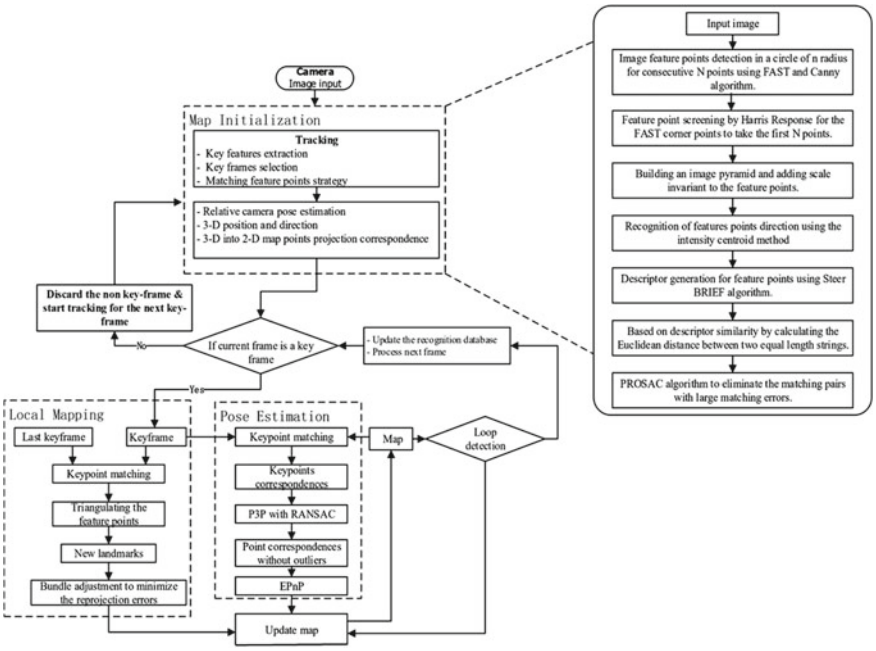


Fig. 1 Flowchart of map construction using Visual-SLAM

2.1.2 Keypoint Reconstruction

A 3D point from consecutive images is calculated using the following equation:

$$P_e = \left(\frac{b(x_1 + x_r)}{2(x_1 - x_r)}, \frac{by}{(x_1 - x_r)}, \frac{bf}{(x_1 - x_r)} \right)^T \tag{1}$$

where, b indicates baseline, and f is the focal length, $y = y_1 = y_r$, while (x_1, y_1) represents the points on one image, and (x_r, y_r) represents the point on the consecutive next image. We set $u = (x_1, y_1, x_r, y_r)$ and $P_c = S(u)$, and therefore, the covariance of the edge point (P_c) is calculated as

$$\sum P_e = \frac{\delta S}{\delta u} \sum u \frac{\delta S^T}{\delta u} \tag{2}$$

Now, we assume $(\sum_u = \text{diag}\sigma_{x1}^2, \sigma_{y_r}^2, \sigma_{y_r}^2)$ and, for the implementation, we take $\sigma_{x_r} = \sigma_{y_1} = \sigma_{y_r} = 0:5[Pixels]$. The correlation between σ_{y_1} and σ_{y_r} is assumed to be very strong.

2.1.3 Camera Motion Estimation

The trajectory of the camera can be estimated by successfully matching the points from time t-1 to t when the points are reconstructed in frame $I_t - 1$, and the points are detected at frame I_t. Let v_t be a camera pose at time t, where P^i_{t-1} is a i-th reconstructed 3D point at t − 1. Similarly, P^i_{t-1} is a point that was taken as a projection of P^i_{t-1} on the image at It. The point P^i_{t-1} is termed a map point because it is stored for map generation, and therefore, point P^i_{t-1} can be represented as $P^i_{t-1} = k(P^i_{t-1}, v_t)$, where k indicates the function of perspective projection:

$$K = N_t^{-1}\left(P^i_{t-1} - M_t\right)$$

$$k\left(P^i_{t-1}, v_t\right) = \left(f\frac{K_x}{K_z}, f\frac{K_y}{K_z}\right)^T \tag{3}$$

where, Mt and Nt are the translation and rotation matrices of vectorv_t. Let g^i_t is a point on the image corresponding to P^i_{t-1}, so the cost function, C can be defined as

$$C(v_t) = \sum_{i=1}^{n} q\left(g^i_t, P^i_{t-1}\right) \tag{4}$$

where $q\left(g^i_t, P^i_{t-1}\right)$ represents the penalty that depends on the Euclidean distance between points g^i_t and P^i_{t-1}. We use the perpendicular distance between the point P^i_{t-1} and the segment containing the point g^i_t in image [37, 38]. We estimate the motion using pose vector v_t at time t, and the correspondence between the points from decreasing cost function $C(v_t)$. This can be achieved by utilizing the gradient descent method, setting the initial value of vector v_t to v_{t-1}, and setting closest point g^i_t to its closest corresponding point, P^i_{t-1}, by calculating the Euclidean distance. This process of point matching repeats, which decreases $C(v_t)$, and the optimal pose vector v_t, and thus, point correspondences are achieved.

2.1.4 Map Construction

We build an incremental 3D map of the environment based on camera pose vector v_t by transforming the 3D points into world coordinates from the camera coordinates. Let us take the camera coordinates and P^i_e as the i-th 3D point, so the location of this point in the camera coordinates can be represented as follows:

$$P^i = c\left(P^i_e, v_t\right) = N_t P^i_e + M_t \tag{5}$$

We integrate the identified 3D points based on their correspondences, which decreases the depth error. Based on the covariance, we integrate the location of all the identified 3D points. We take the average location of the identified 3D points between the keyframes, which increases the efficiency. The created 3D points indicate the map, and estimate the trajectory of the camera, between the keyframes.

2.1.5 Camera Motion Update

Camera motion is updated by extracting the keyframe from the sequence of images with interval d, and then, we refine the motion using the RANSAC algorithm between the keyframes. As expected, the camera motion is relatively large between the keyframes, so to avoid the local minima, we initialize the value of a keyframe to I_d from the estimated camera motion by each keyframe $I_t + 1$. Every 3D point P^i_{t-d} taken upto keyframe $I_{t\text{-}d}$ is supposed to project onto keyframe I_t and match to the 3D point q^i_t in the image [39].

Uncertainty is evaluated by calculating the covariance matrix of camera poses. We use $\overline{v_t}$ and $\sum_{v_t}$ to represent the mean and covariance, in which $\overline{v_t}$ is calculated from the keyframe, whereas $\sum_{v_t}$ is calculated with the following mechanism. Let s_t represents the vector of multiple points in the image at time t where w_t indicates the vector of 3D points, which are matched with s_t. We can indicate s_t as $s_t = h(w_t, v_t) + n_t$, where n_t is noise having zero mean and zero covariance, $\sum_{n_t}$, and s_t can be obtained with the Taylor expansion, as follows:

$$s_t \approx k(\overline{w_t}, \overline{v_t}) + \frac{\delta k}{\delta w_t}(w_t - \overline{w_t}) + \frac{\delta k}{\delta v_t}(v_t - \overline{v_t}) + n_t \tag{6}$$

We can calculate the covariance of camera trajectory utilizing Eq. 6, as follows:

$$\sum v_t = \left(J^T_{v_t} \left(\sum\nolimits_{n_t} + J_{wt} \sum\nolimits_{wt} J^T_{wt} \right)^{-1} J_{vt} \right)^{-1} \tag{7}$$

where, $J_{vt} = \frac{\delta k}{\delta v_t}(\overline{w_t}, \overline{v_t})$, $J_{wt} = \frac{\delta k}{\delta wt}(\overline{w_t}, \overline{r_t})$ and $\sum_{wt}$ represents the covariance matrix of the 3D points that match s_t. The size of the $\sum_{wt}$ depends on the number of 3D points and if the number of points is large, which makes $\sum_{wt}$ computation intractable.

We assume that the location of all the 3D points that are reconstructed from the same frame have a strong correlation. To overcome the complexity, we divide all 3D points into two parts, w_a and w_b, where w_a indicates the 3D points reconstructed from the last keyframe, $I_{t\text{-}d}$, and w_b represents the reconstructed 3D points from the past key frames, I_1 to $I_{t\text{-}2d}$. we can approximate each group covariance to the mean covariance of all 3D points. Considering all the assumptions, we have the following:

$$J_{wt}\sum\nolimits_{wt}J_{wt}^{T}=\frac{1}{|w_a|}\sum_{P\in w_a}J_P\sum\nolimits_{P}J_P^{T}+\frac{1}{|w_b|}\sum_{P\in w_b}J_P\sum\nolimits_{P}J_P^{T}$$

$$J_P=\frac{\delta k}{\delta P}\left(\overline{P},\overline{v_t}\right) \tag{8}$$

where, J_P and $\sum_P$ represent the Jacobian and covariance matrix of a 3D point, respectively. This supposition decreases the computational complexity of the system.

2.1.6 Map Update

We construct the 3D map according to section II-A4. We fuse the matched 3D points with weights according to their covariance. The 3D point explained in section II-A4 can also be expressed as

$$P_t^i=c\left(P_{e,t}^i,v_t\right) \tag{9}$$

As mentioned above, we are ignoring correlation term $\sum_{wt}$, and therefore, we calculate the covariance matrix of each 3D point. Let $\overline{P_t^i}$ and $\sum_{P_t^i}$ represent the mean and covariance of a 3D point, respectively. Using the Taylor expansion, we have the following:

$$P_t^i\approx c\left(\overline{P_{e,t}^i},\overline{v_t}\right)+\frac{\delta c}{\delta P_{e,t}^i}\left(P_{e,t}^i-\overline{P_{e,t}^i}\right)+\frac{\delta c}{\delta v_t}(v_t-\overline{v_t}) \tag{10}$$

The covariance of 3D point P_t^i can be calculated using Eq. 10 as follows:

$$\overline{\sum\nolimits_{P_t^i}}=\frac{\delta c}{\delta P_{e,t}^i}\sum P_{e,t}^i\frac{\delta c}{\delta P_{e,t}^i}^{T}+\frac{\delta c}{\delta v_t}\sum v_t\frac{\delta c}{\delta v_t}^{T} \tag{11}$$

We update the location and covariance of a 3D point by fusing Eq. 11 with the point at t−d, as follows:

$$\overline{P_t^i}=\overline{P_{t-d}^i}+\sum P_{t-d}^i\left(\sum\nolimits_{P_{t-d}^i}+\overline{\sum\nolimits_{P_t^i}}\right)^{-1}\left(P_t^i-\overline{P_{t-d}^i}\right)$$

$$\sum P_t^i=\left(\sum\nolimits_{P_{t-d}^i}^{-1}+\overline{\sum\nolimits_{P_t^i}^{-1}}\right)^{-1} \tag{12}$$

3 Swarm-Based Path Planning Approach

In this section, we introduce the proposed path planning scheme based on particle swarm optimization. The elevation map generated by the visual-SLAM algorithm is used as input terrain information for the optimization algorithm to plan the optimum path. The data set we used in our system is very diverse, and provides information on the terrain. There are multiple system constraints, which must be satisfied before planning the path from source to destination and meeting the multiple objectives we desire in order to obtain the maximum value. In this regard, we propose the DFF to derive the optimal trajectory of the UAVs while considering all the constraints and objectives of the system.

3.1 Working Principle of Particle Swarm Optimization

PSO is a heuristic search algorithm. It was first developed by Kennedy and Eberhart in 1995 to introduce a method for optimization of a nonlinear function [40]. It is a nature-inspired set of computational methodologies to resolve complex real-world problems. PSO computes the number of particles to look for the best solution. Each particle moves in accordance with both its previous best particle in the group and the swarm's global best particle. Each particle changes its velocity and location in real time using information from the prior velocity and best position obtained by any particle in the group, as well as the global swarm's best position.

3.2 PSO Formulation

The mathematical formulation for each particle's velocity and position are stated as follows. Let the total number of particles in a swarm be P, the total iterations is N, and the 3D dimension of each particle is D. Therefore, for the particle, position x and velocity v can be represented as:

$$\begin{aligned} x_i &= (x_{i1}, x_{i2}, \ldots, x_{iD}) \\ v_i &= (v_{i1}, v_{i2}, \ldots, v_{iD}) \end{aligned} \tag{13}$$

The position for the best particle, $p_{i,best}$, in the group and the global best swarm paticle, s_{best}, can be computed as follows:

$$\begin{aligned} p_{i,best} &= \left(p_{i1,best}, p_{i2,best}, \ldots, p_{iD,best}\right) \\ s_{best} &= \left(s_{1,best}, v_{2,best}, \ldots, v_{D,best}\right) \end{aligned} \tag{14}$$

Because p_{best} and s_{best} are termed cost values for PSO, once a cost function is defined, then the position and velocity are updated as follows:

$$\begin{aligned} x_{ij}^{t+1} &= x_{ij}^{t} + v_{ij}^{t+1} \\ v_{ij}^{t+1} &= \chi v_{ij}^{t} + ar_1\left(p_{i,j,best} - x_{ij}^{t}\right) + b_{r_2}\left(s_{j,best} - x_{ij}^{t}\right) \\ \text{For i} &= 1, 2, 3, \ldots \text{ P j} = 1, 2, 3 \ldots \text{D t} = 1, 2, 3, \ldots \text{N} \end{aligned} \tag{15}$$

where, a and b are the self-cognitive acceleration property and the social knowledge parameter of the swarm, respectively, which represent the inheritance characteristics of the personal particle and the whole swarm; and are random values in the range [0–1], and χ represents the inertia of an individual particle, which induces an effect on the velocity from one iteration to next iteration. The authors in [41] suggested optimum values of a = b = 1:496 and χ = 0:7298 for PSO performance.

4 Proposed Dynamic Fitness Function

In order to derive the optimal trajectories, the DFF computes the fitness of the trajectory considering optimization parameters, which are divided into two groups, namely, objectives and constraints. The former consist of risk estimation, energy consumption, and operation completion time; the latter are further divided into two parts depending upon the UAV's physical constraints (flying slope and turning angle) and the physical limitations of the environment (region sensitivity, restricted areas, and terrain constraints). The working flow of the DFF can be observed in Fig. 2. The DFF can be formulated as seen in equation:

$$DFF_{fitness} = F_{objectives} + F_{constraints} \tag{16}$$

where, $F_{objectives}$ indicates the objectives function on which we focus to gain the maximum value, whereas $F_{constraints}$ indicates the UAV physical and environmental restrictions, which must be fulfilled before planning the trajectory.

4.1 Objectives Design

We have set optimization parameters, and the objectives were constructed to improve the quality of path planning. The objectives can be represented as weighted components of risk estimation, energy consumption, and operation completion time, as follows:

$$F_{objectives} = w_1 O_{RE} + w_2 O_{EC} + w_3 O_{OT} \tag{17}$$

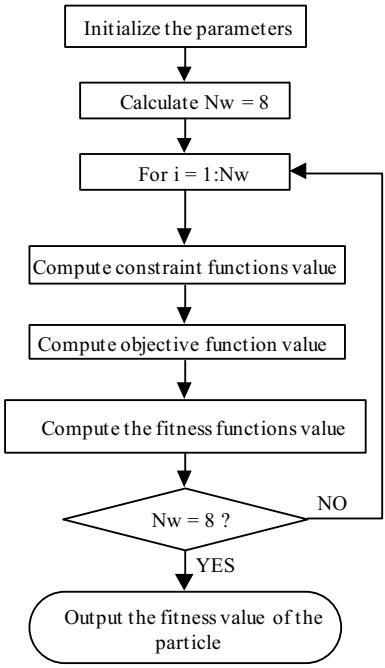


Fig. 2 Flowchart to compute dynamic fitness function

where, w_1, w_2, w_3 denote the weights of the objective components [39], which are chosen to derive the importance of each component while planning the path, and O_{RE}, O_{EC}, O_{OT} are functions from which values are taken in the range [0, 1]. We aim to derive the optimum path with less risk, energy, and time.

4.1.1 Risk Estimation

Some flight restrictions should be implemented. In harsher weather conditions, such rain, snow, or strong winds, small UAVs are susceptible to damage. The UAV altitude while doing the work should be moderate; winds at higher altitudes are stronger. The UAV also faces risks because of dense clouds that impede its ability to focus. Based on the above risks, we identify the following two types of risk.

1. Environmental Risk

The environment has a wide range of characteristics, and therefore, it is difficult to make a model that precisely measures the environmental risk. Therefore, for simplicity, an environmental value is generated randomly, $r^e_{i,i+t}$, that represents the risk from the i-th waypoint to waypoint (i + 1). The summation of the risk values would be considered the environmental risk.

2. Altitude Risk

The altitude risk is actually an absolute difference in altitude between two waypoints, and therefore, we formulate altitude risk $r^a_{i,i+t}$ as follows:

$$r^a_{i,i+t} = k * (z_{i+1} - z_i) \tag{18}$$

where, k represents a constant parameter for control. Because risk analysis depends on location, it will change according to weather conditions and the UAV's altitude at the same instant during flight. Therefore, the total risk can be formulated as follows:

$$O_{RE} = \frac{\sum_{i=1}^{N_{w-1}} RE_i}{maxRE} \tag{19}$$

$$RE_i = w_E Rr^e_{i,i+t} + w_{AR} r^a_{i,i+t} \tag{20}$$

RE_i shows the total risk from the i-th waypoint to waypoint (i + 1), while w_{ER} and w_{AR} are the weight factors of the environmental and altitude risks, respectively. N_w denotes the total number of waypoints from source to destination, and $maxRE$ is a normalized value of the risk, which can be computed as follows:

$$maxRE = (N_w - 1) * \left[w_{ER} * Z * w_{AR}\left(2 * maxr^e\right)\right] \tag{21}$$

where, $maxr^e$ indicates the maximum value instigated by the environment risk, and Z is the altitude of the UAV during flight.

4.1.2 Energy Consumption

Fuel is essential to UAV missions. If the UAV does not arrive on time, the mission is said to have failed. A simple method that uses less energy (EC) should be the priority. We assume the UAV velocity stays constant during flight. We define EC as follows:

$$O_{EC} = \frac{\sum_{i=1}^{N_w-1} FC_i}{maxFC} \tag{22}$$

$$FC_i = P_u * t_{i,i+1} \tag{23}$$

$$t_{i,i+1} = \frac{d_{i,i+1}}{v} \tag{24}$$

$$d_{i,i+1} = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{25}$$

where, FC_i represents the fuel burned in flying from the i-th waypoint to waypoint (i + 1). Pu is the power of the UAV at velocity v, while $t_{i,i+1}$ is the total time taken by the UAV to fly from the i-th waypoint to waypoint (1 + i); $d_{i,i+1}$ indicates the Cartesian distance of a flight from the i-th waypoint to waypoint (1 + i), and $maxFC$ is a normalized value for fuel consumption, which can be formulated as follows:

$$maxFC = (N_w - 1) * P_u * \frac{d_{max}}{v} \tag{26}$$

where, $d_{max} = \sqrt{X^2 + Y^2 + Z^2}$ where X, Y, Z indicate the three dimensions of the UAV, i.e., the X-axis, Y-axis, and Z-axis, respectively, during flight time.

4.2 Constraints Design

To optimize possible flight paths Constraints are 0 when satisfied, otherwise a penalty is applied. Applying a penalty Q assures that the path from source to destination is always feasible. Considering the physical restrictions on the UAV and the environment's limits, we can formulate the constraints as follows:

$$F_{Constraints} = UAV_{constraints} + Environment_{constraints} \tag{27}$$

4.2.1 UAV Constraints

The UAVs have physical properties that cause these constraints. The UAV's behavior during maneuvering should be treated as a priority, as it offers smoothness in flight. In this regard, we care for the most crucial aspects of a UAV: slope and rotation. UAV limitations are therefore defined as follows:

$$UAV_{constarints} = TA + FS \tag{28}$$

1. Turning Angle

The turning angle indicates a UAV's maneuverability in the horizontal direction, i.e. the angle taken during flight from the previous and current directions. The turning angle should be less than the maximum tolerable threshold for turning, thus we calculate it as follows:

$$TA = 0, TA = \sum_{i=2}^{N_w - 1} TA_i$$

where,

$$TA_i = \begin{cases} Q, if\theta > \theta_{max} \\ 0, otherwise \end{cases} \tag{29}$$

where, θ defines the turning angle of the UAV in 3D directions (xi, yi, zi), and $\theta_{\max}$ maximum tolerable angle. The authors in [34] provided the formulation to calculate turning angle θ_i as follows:

$$\theta = \arccos\left(\frac{(p_{xi}, p_{yi})(p_{xi+1}, p_{yi+1})^T}{\|p_{xi}, p_{yi}\|_2 \|p_{xi+1}, p_{yi+1}\|_2} \right) \tag{30}$$

where, $p_{x_i} = x_i - x_{i-1}$, $P_{x_{i+1}} = x_{i+1} - x_i$, $p_{y_i} = y_i - y_{i-1}$, $P_{y_{i+1}} = y_{i+1} - y_i$ and $\|x\|_2$ is a vector norm for a vector x.

2. Flying Slope

The flying slope is defined as the mobility of a UAV while gliding and while climbing. During flight, the UAV's slope is along the horizontal from one waypoint to the next. Given the permissible gliding and ascending angles, the slope of a UAV is derived as:

$$FS = 0, FS = \sum_{i=2}^{N_w} FS_i$$

where,

$$FS_i = \begin{cases} Q, if f_i \notin [\tan(\alpha_{max}), \tan(\beta_{max})] \\ 0, otherwise \end{cases} \tag{31}$$

where, FS_i is the flying slope from one waypoint to the i-th waypoint; α_{max} and β_{max} represent the maximum tolerable gliding and climbing angles, and f_i can be formulated, according to [34], as follows:

$$f_i = \frac{z_i - z_{i-1}}{\|x_i - x_{i-1}, y_i - y_{i-1}\|_2} \tag{32}$$

where, f_i is the flying slope taken by the UAV from the i-th waypoint (x_i; y_i; z_i).

4.2.2 Environment Constraints

Due to the external environment, the UAV must follow specific rules. Restricted areas such as military sites, key government institutions, etc., should be taken into consideration. Therefore, a system should be planned to avoid these limited locations. Likewise, unforeseen events can occur in which the UAV encounters a flying toy, for example, an unregistered aerial vehicle, or birds in flight. Regional sensitivity is used to handle these types of circumstances. This deals with randomly generated sensitive regions where the UAV recognizes a threat and computes a safe path to avoid them. Furthermore, the terrain restricts flight. Environmental constraints can be expressed as follows. We divide the path into a 20 × 20 grid, and the UAV can sense four cells around itself.

$$Environment_{constraints} = RA + RS + TL + ML + CA \tag{33}$$

1. Restricted Area

There are some specific areas that UAVs are not permitted to fly through due to restrictions, such as cantonments, restricted government territories, and so on, and hence the feasible path to the destination should be a legal one that avoid those regions. For simplicity, we consider a restricted area to be a rectangle. Formulation of a restricted region as follows:

$$RA = 0, RA = \sum_{i=1}^{N_w} RAC_i \tag{34}$$

where,

$$RAC_i = \begin{cases} Q, if\ waypoint\ in\ Range(x_r, y_r) \\ 0, otherwise \end{cases}$$

where, $Range(x_r, y_r) = \{m_x \leq x_r \leq n_x\} \cap \{m_y \leq y_r \leq n_y\}$ and m_x and n_x represent the lower and upper bounds, respectively, for x coordinates of the r-th restricted area at the i-th waypoint, whereas m_y and n_y indicate the lower and upper bounds, respectively, of y coordinates of the r-th restricted area at the i-th waypoint.

2. Region Sensitivity

Unconditional and unexpected events might occur during flight. Thus, all hazardous events in the path of a UAV are randomly generated. It notices the hazard and constructs a path to avoid them. This can be formulated as follows:

$$Rs = 0, Rs = \sum_{i=1}^{N_w} Rs_i(t) \tag{35}$$

where

$$Rs_i = \sum_{cellx \in N(i)} v_{cellx}(t)$$

$$Rs_i = \begin{cases} Q, if\, Rs_i > Rs_{th} \\ 0, otherwise \end{cases} \tag{36}$$

where, $Rs_i(t)$ is the value for sensitivity at the i-th waypoint during the flight at time t, and $v_{cellx}(t)$ is the cell value at flight time t. N(i) is the set of neighbor cells for the i-th waypoint. The UAV checks the values of the cells at every waypoint, and if any cell has a sensitivity value greater than the threshold, penalty Q will be given. It checks the values of the set of neighbor cells for N(i) to avoid that region to satisfy the constraint.

3. Terrain Limits

During flight, a UAV should take into consideration the limitations of the terrain so that the UAV always flies above it and avoids collisions (for example, with mountains). To adhere to a terrain constraint, the algorithm gives penalty Q to provide the feasible path. This constraint can be formulated as follows:

$$TL = 0, TL = \sum_{i=1}^{N_w} TL_i \tag{37}$$

where,

$$TL_i = \begin{cases} Q, if\, z_i \leq map(x_i, y_i) \\ 0, otherwise \end{cases}$$

where, $map(x_i, y_i)$ is a function that returns the altitude of the terrain location at point (x_i, y_i), which finds the number of points inside that location.

4. Map Limits

For a feasible path, the UAV must stay inside the mission space to avoid uncertainties. Therefore, the algorithm applies penalty Q to the points of a trajectory that are off the map limits. This constraint ensures the space of a mission can be formulated as follows:

$$ML = \sum_{i=1}^{N_w} ML_i \tag{38}$$

where,

$$ML_i = \begin{cases} 0, Inmap(x_i, y_i) \\ Q, Otherwise \end{cases}$$

$$Inmap(x_i, y_i) = \left(x_l^m \leq x_i \leq x_u^m\right) \Lambda \left(y_l^m \leq y_i \leq y_u^m\right) \tag{39}$$

where, x_l^m and x_u^m are the lower and upper bounds, respectively, for the x coordinate, and y_l^m and y_u^m are the lower and upper bounds, respectively, for the y coordinate. The minimum value to satisfy the map constraint is ML = 0.

5. UAV Collision Avoidance

When calculating paths for multiple UAVs, the planner must ensure that the UAVs do not get too close to each other, increasing the possibility of a collision while following their individual paths. To keep a safe distance between them, the limitation can be expressed as follows:

$$CA = \sum_{i=1}^{N_w^u} \sum_{j=1}^{N_w^u} CA_i \tag{40}$$

where,

$$CA_i = \begin{cases} Q, if\, d_{ij}^{uv} < d_{min} \\ Q, otherwise \end{cases}$$

$$d_{min}^{uv} = \sqrt{\left(x_i^u - x_j^v\right)^2 + \left(y_i^u - y_j^v\right)^2 + \left(z_i^u - z_j^v\right)^2} \tag{41}$$

where, d_{min} is the minimum distance between the UAVs to avoid a collision, and d_{ij}^{uv} is the distance between the i-th waypoint and the j-th waypoint of the u-th UAV trajectory and the j-th UAV trajectory, respectively.

5 Operation of the Proposed Path Planner

In this section, we explain the working mechanism of the proposed multiple UAV–path planner, which is based on visual-SLAM, PSO, and the DFF explained in Sects. 2, 3, and 4, respectively. The proposed planner first utilizes the elevation map generated by visual-SLAM and fed into the PSO planning algorithm to derive the optimum trajectory for each UAV to the defined destinations, in which the DFF optimizes all the possible waypoint sequences to reach destinations considering all constraints and objectives, along with satisfying the collision avoidance condition. If all conditions are satisfied, the planner will output the optimum trajectory for each UAV to its destination.

In our proposed system, the path from source to destination consists of waypoints and line segments. We opted for an eight-waypoint trajectory-generation system. For clear understanding, we divided the whole operation area into cells and determine the estimated flight time to the destination. Next, we initialize the PSO algorithm to plan the optimum path for each UAV, which can be seen in Fig. 3 from step 5–33. In the quest to attain the optimum trajectory for each UAV, at first, the planner randomly generates the velocity and position vectors of particle PN. Next, using Eq. (15), the velocity and position vectors of each particle are updated.

After that, the proposed DFF is applied to the updated particle as shown the working flowchart of the DFF in Fig. 2. Considering all the constraints and objectives,

Algorithm 1 Pseudocode of proposed UAVs Path Planner

```
1:    Set flight time =T_flight ;
2:        Initialize the cell values;
3:      for i = 1: T_flight
4:      {
5:        while (CA is not satisfied)
6:        {
7:          Set UAV number= N ;
8:            for j = 1: N
10:             Set iteration number= N_iter ;
11:             Set particle number= P_n ;
12:               for k =1: N_iter
13:               {
14:                  for t =1: P_n
15:                  {
16:                     Randomly initialize x_t and v_t ;
17:                     Initialize P_t,best,= x_t , S_best = x_Pn ;
18:                     Update x_t and v_t using the updating formula (15);
19:                     Compute the fitness value of x_t using formulas (16) -(39);
20:                     if ( fitness (x_t) > fitness (P_t, best))
21:                     { P_t,best = x_t ;}
22:                     if ( fitness (P_t,best) > fitness (S_best ) )
23:                     {
24:                        S_best = P_t,best ;
25:                        Opt _fitness = fitness (S_best );
26:                      }
27:                     Sense the sensitive region using formula (36);
28:                        if  (RS(i)<RS(th))
29:                          Opt _fitness = fitness (S_best );
30:                        else
31:                            return to step 19;
32:                   }
33:                }
34:                Evalution collision among multi-UAVs using formula (40);
35:         }
36:      Output the collision-free trajectories for multiple UAVs
37:      }
38 :    Output T_flight flight time path planning results;
```

Fig. 3 Pseudocode of the proposed path planner

the DFF optimizes each particle and finally outputs the best particle, pibest and the global best particle in the swarm, sbest, which is explained in Sect. 4. The DFF output is based on the fitness value acquired by each particle. Then, we store the optimum path for the first UAV and set the iteration number to Nt. Before initializing the other UAVs, we aim to derive a collision-free path, and therefore, we check collision avoidance condition CA. If CA is satisfied, the planner outputs optimum trajectories for all UAVs; otherwise, it goes back to step 5 if the CA is not satisfied. Finally, when the flight time reaches, the planner will output the optimum paths for all UAVs to their respective destinations. The process of the proposed planner is represented in the pseudocode algorithm shown in Fig. 3.

6 Simulation Results

In this section, we develop a Matlab-based operational environment to evaluate the working performance of the proposed two-step UAV path–planning system. The main simulation parameters are listed in Table 1. In our implementation, we used a data set [42] that was collected by a monocular camera installed at the quad-copter, in different environments. The data set is publicly available, and more details can be found at midair.ulg.ac.be. The data set was utilized as input to the optimization algorithm for multiple-UAV path planning algorithm. We used different types of test sequences, which can be seen in Figs. 4 and 5 in our system to construct an online map of the environment. Figure 6a indicates the features in the consecutive scenes that were matched to simultaneously build an incremental map, which can be seen in Fig. 6b. The points cloud map contains information on the x, y, z positions and normal at every point. The terrain representations from the points cloud can be seen in Fig. 7. We utilized a triangulation algorithm [43] to reconstruct the terrain from

Table 1 Simulation parameters

Parameter	Value
No. of UAVs	2
Speed	10 m/s
Power	20
Iteration number	32, 64, 128, 256, 512
Sensitivity threshold	10
Turning angle threshold	85°
Gliding angle threshold	−30°
Climbing angle threshold	30°
Minimum distance threshold	0.2
Initial environmental risk	1–5
Flight time threshold	2
Grid size	20 × 20

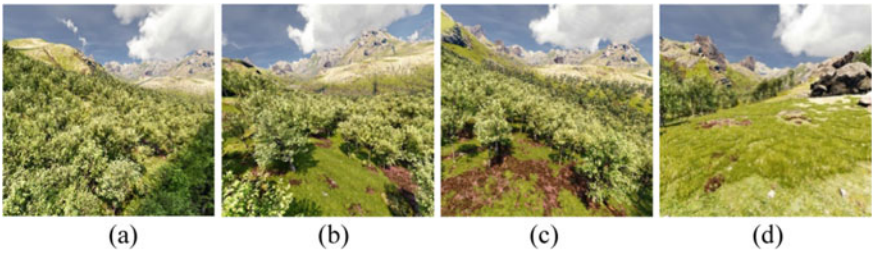


Fig. 4 Flight path (sequence 0005)

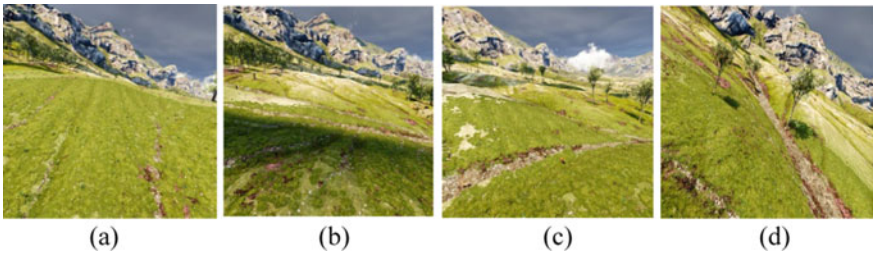
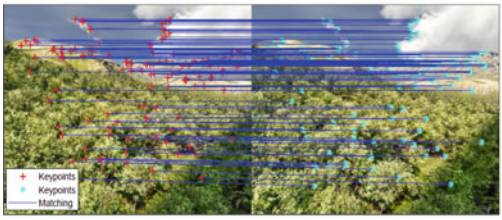
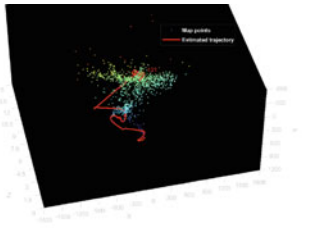


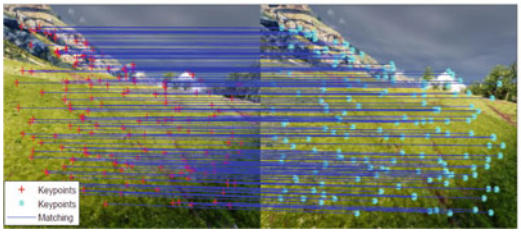
Fig. 5 Flight path (sequence 0012)



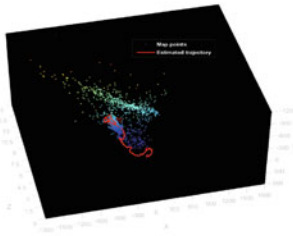
(a) Matching Features (sequence 0005)



(b) Points cloud Map (sequence 0005)



(a) Matching Features (sequence 0012)



(b) Points Cloud map (sequence 0012)

Fig. 6 **a** Image registration between consecutive scenes and **b** Map construction by VSLAM to be used for path planning

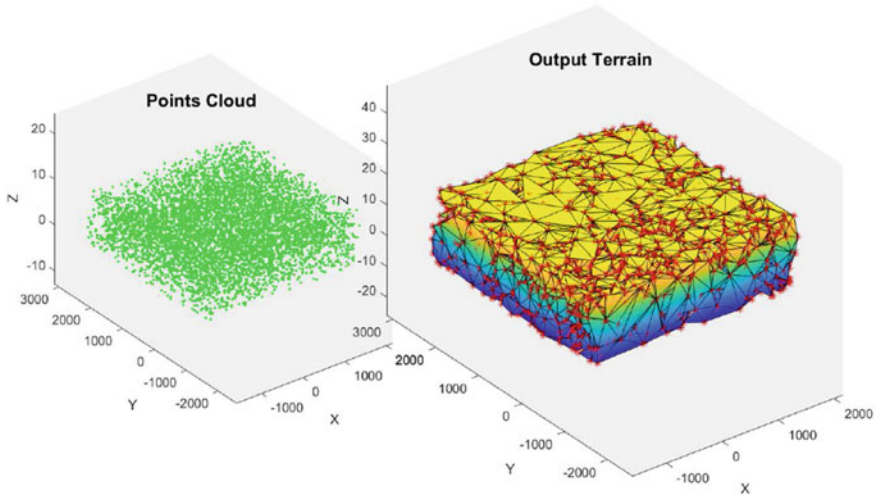


Fig. 7 Terrain representation from the points cloud of sequence 0012

the points cloud.

Figure 8 shows the effect of different numbers of particles on the optimal fitness value of the proposed DFF. We can clearly see that the fitness value of the proposed DFF converges to a stable value faster as the number of particles and iterations

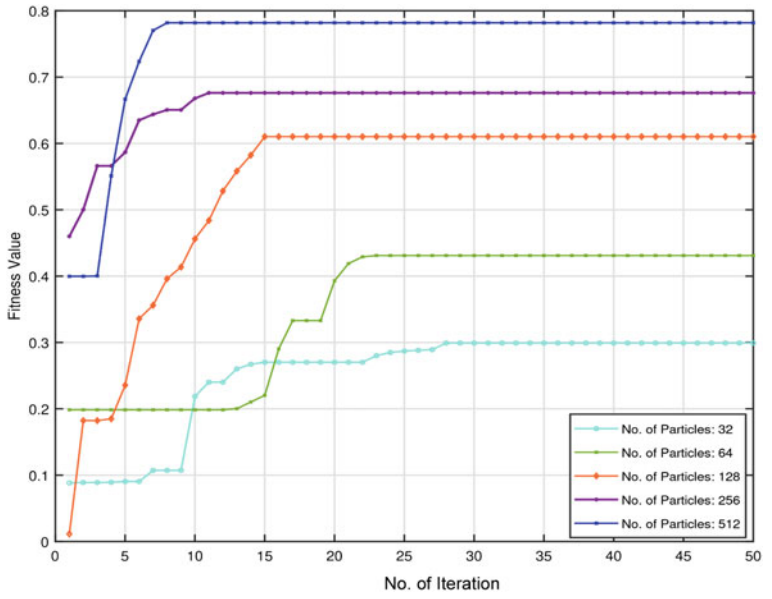
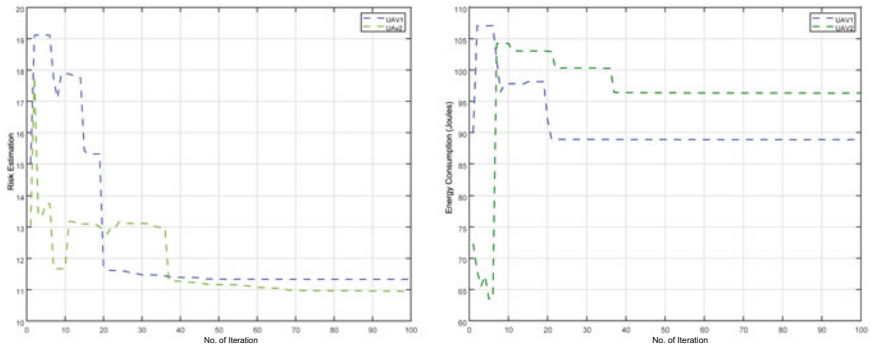


Fig. 8 Optimal fitness values for different numbers of optimization particles



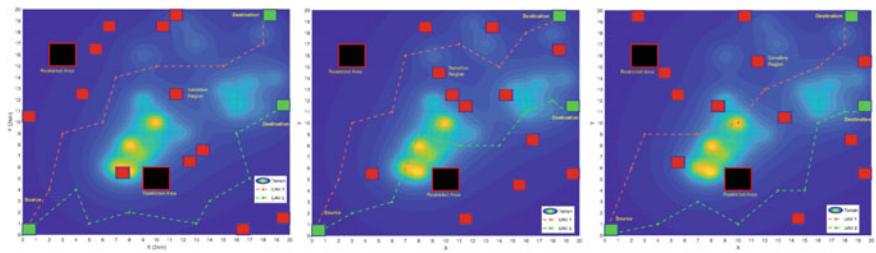
(a) (b)

Fig. 9 Optimization of the PSO path planner performance in terms of **a** risk estimation and **b** energy consumption

increases. The optimization performance of the path planner in terms of energy consumption and flight risk estimation can be observed in Fig. 9. We utilized 128 particles in our system. As the number of iterations increased, the values of energy consumption and flight risk estimation converged to a stable value. Moreover, the difference between the optimum value of energy consumption, where both UAVs converge, is less than five, and the values of flight risk estimation for both UAVs is similar, which depicts the effectiveness of the proposed path planner by ensuring fairness between the generated paths for both UAVs.

Figure 10 shows the optimal paths followed by UAV 1 and UAV 2 from source to destination while avoiding sensitive regions and restricted areas, respectively, for the first three flights. The small red 1×1 rectangles have a sensitivity greater than the threshold, while the black 2×2 rectangles indicate restricted areas where UAVs are not allowed to fly.

The sensitive regions generate randomly, indicating a hazardous event, so the proposed planner optimizes the path until hazardous free paths to the destinations



(a) First Flight (b) Second Flight (c) Third Flight

Fig. 10 Optimal trajectories of the UAVs from source to destination using proposed algorithm

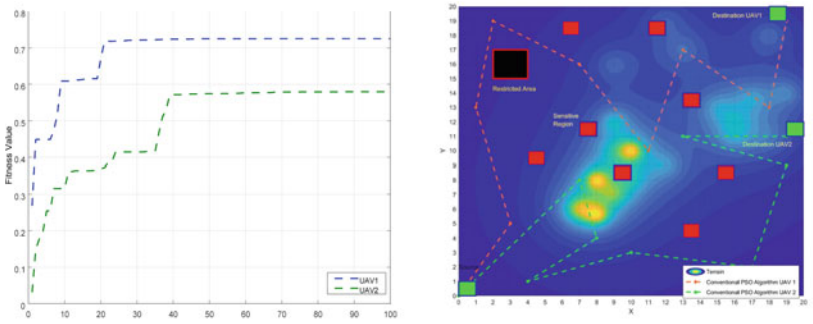


Fig. 11 **a** Optimal fitness values attained and, **b** UAVs Flight using Conventional PSO Algorithm

are determined. We can observe that the trajectories generated for each flight time avoids all the sensitive regions and reach the destination safely.

The proposed algorithm also ensures that the multiple UAVs do not collide with each other. The green 1×1 rectangles represent the source and destination. In Fig. 10, the yellow highlighted areas are high elevations. We can also see that the trajectory waypoints generated do not overlap, and a UAV reaches the destination by following the shortest path, which indicates the high efficiency of the proposed planner. Therefore, Fig. 11a indicates the high fitness value attained by each UAV driven by the proposed path planner.

Figure 11b indicates the trajectories generated by the conventional PSO. As the defined environment is dynamically complex due to which conventional PSO is incompatible with adapting the situation; therefore, it takes very high computational time to converge. Considering the incompatibility of the conventional PSO in our environment, we choose to make the environment less complicated and convenient to converge. The computational time for the conventional PSO for the simple environment is higher than our proposed algorithm in the dynamic and complex environment. The conventional PSO takes 1,767 s while our proposed algorithm takes 739.8 s.

The same computer is used to run the both algorithms. The Table 2 indicates the flight statistics of both algorithms for the first flight. The conventional PSO algorithm for both UAVs reaches the destination following the long path. It takes more travel time while our proposed algorithm reaches the destination for both UAVs following the shortest path and in optimal travel time in a highly complex environment. The distance covered from one waypoint to another and the corresponding flight times for both UAVs can be seen in Fig. 12a, b.

The total distances from the source to destination covered by UAVs during the first flight were 3,062.4369 m and 3,065.0706 m. Likewise, the times taken to reach the destinations for both UAVs were almost the same i.e., 307 s. Similarly, Fig. 12c, d indicates the distance covered and corresponding flight time for both UAVs from one waypoint to another using the conventional PSO algorithm. We can observe that the distance and time taken at each waypoint is greater than the proposed algorithm. The total distance covered by the UAVs for the first flight is 5,571.9591 (m) and

Table 2 Distance and Time comparison with the conventional PSO

Parameter	Value
(a) Flight dynamics of first flight using proposed algorithm	
Distance covered by UAV 1	3,062.4369 (m)
Distance covered by UAV 2	3,065.0706 (m)
Travel time by UAV 1	307.2542 (s)
Travel time by UAV 2	307.4481 (s)
(b) Flight dynamics of first flight using conventional PSO algorithm	
Parameter	Value
Distance covered by UAV 1	3,062.4369 (m)
Distance covered by UAV 2	3,065.0706 (m)
Travel time by UAV 1	307.2542 (s)
Travel time by UAV 2	307.4481 (s)

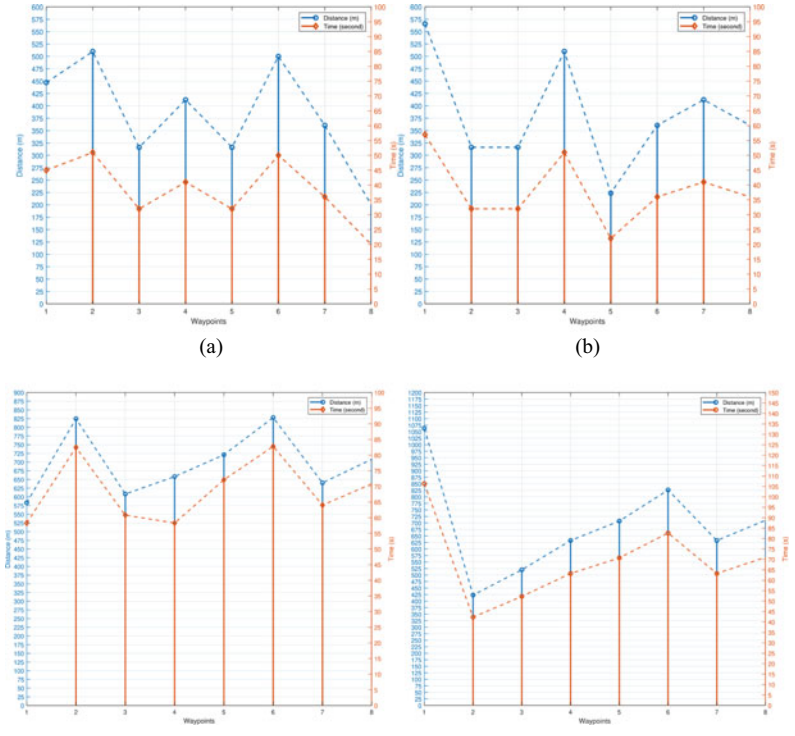


Fig. 12 UAV flight dynamics in terms of distance and time using proposed algorithm for **a** UAV 1 and **b** UAV 2 and using conventional PSO algorithm **c** UAV 1 and **d** UAV 2

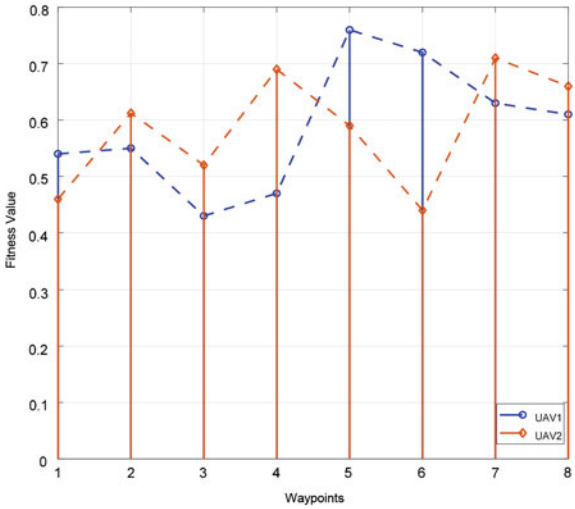


Fig. 13 Fitness values attained at each waypoint during the flight by proposed algorithm

6,065.0706 (m). Similarly, the total time consumed by UAV1and UAVs is 551.6796 (s) and 549.7633 (s), respectively. In Fig. 13, we show the fitness values attained at each waypoint by both UAVs during their flights. We sum up the optimal fitness values of all waypoints for UAV 1 and UAV 2. The total optimal fitness for all the waypoints of UAV 1 and UAV 2 were 5.52 and 5.51, respectively, which are virtually the same and which depict the fairness of our proposed twostep path planner.

7 Conclusions

In this paper, we designed a two-step, centralized system to construct a map using state-of-the-art visual-SLAM. We introduce corner-edge points matching mechanism to stabilize the system with the least extracted map points. The proposed algorithm effectively detects the keypoints in different environments and successfully registered the features. The constructed map is processed as an input mean for the particle swarm optimization algorithm to plan UAVs' optimum path. We proposed a dynamic fitness function considering different optimization objectives and constraints in terms of UAV flight risk estimation, energy consumption, and maneuverability for the operational time. We also proposed a path updating mechanism based on region sensitivity to avoid sensitive regions if any hazardous and unexpected event detects in UAVs' paths. The system effectively avoids the sensitive regions and returns collision-free paths to reach UAV to the destinations safely. The simulation results validate the effectiveness of our proposed PSO-VSLAM system. We currently consider two UAVs over different flight times to evaluate our proposed PSO-VSLAM system's performance, and it successfully outputs the collision-free trajectories and proves high adaptability towards the complex dynamic environment. Therefore, we plan to consider more than

two UAVs in our future work and implement machine learning algorithms because our proposed system effectively achieves the collision-free trajectories for two UAVs while adapting to the highly dynamic and complex environment.

Acknowledgement This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant NRF-2019R1F1A1061696.

References

1. Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I, Leonard JJ (2016) Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans Rob 32(6):1309–1332
2. Trujillo JC, Munguia R, Guerra E, Grau A (2018) Cooperative monocular-based SLAM for multi-UAV systems in GPS-denied environments. Sensors 18(5):1351
3. Du H, Wang W, Xu C, Xiao R, Sun C (2020) Real-time onboard 3D state estimation of an unmanned aerial vehicle in multi-environments using multi-sensor data fusion. Sensors 20(3):919
4. Ramezani M, Tinchev G, Iuganov E, Fallon M (May 2020) Online LiDAR-SLAM for legged robots with robust registration and deep-learned loop closure. In: 2020 IEEE international conference on robotics and automation (ICRA), pp 4158–4164
5. Stentz A, Fox D, Montemerlo M (2003) Fastslam: a factored solution to the simultaneous localization and mapping problem with unknown data association. In: Proceedings of the AAAI national conference on artificial intelligence
6. Loo SY, Mashohor S, Tang SH, Zhang H (2020) DeepRelativeFusion: dense monocular SLAM using single-image relative depth prediction. arXiv:2006.04047
7. Shakhatreh H, Sawalmeh AH, Al-Fuqaha A, Dou Z, Almaita E, Khalil I, Othman NS, Khreishah A, Guizani M (2019) Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges. IEEE Access 7:48572–48634
8. Mughal UA, Xiao J, Ahmad I, Chang K (2020) Cooperative resource management for C-V2I communications in a dense urban environment. Veh Commun 26:100282
9. Mughal UA, Ahmad I, Chang K (2019) Virtual cells operation for 5G V2X communications. In: Proceedings of KICS, pp 1486–1487
10. Shakoor S, Kaleem Z, Baig MI, Chughtai O, Duong TQ, Nguyen LD (2019) Role of UAVs in public safety communications: energy efficiency perspective. IEEE Access 7:140665–140679
11. Wen S, Zhao Y, Yuan X, Wang Z, Zhang D, Manfredi L (2020) Path planning for active SLAM based on deep reinforcement learning under unknown environments. Intell Serv Robot 1–10
12. Kalogeiton VS, Ioannidis K, Sirakoulis GC, Kosmatopoulos EB (2019) Real-time active SLAM and obstacle avoidance for an autonomous robot based on stereo vision. Cybern Syst 50(3):239–260
13. Doitsidis L, Weiss S, Renzaglia A, Achtelik MW, Kosmatopoulos E, Siegwart R, Scaramuzza D (2012) Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. Auton Robot 33(1):173–188
14. Alzugaray I, Teixeira L, Chli M (May 2017) Short-term UAV path-planning with monocular-inertial SLAM in the loop. In: 2017 IEEE international conference on robotics and automation (ICRA), pp 2739–2746
15. Sánchez-García J, Reina DG, Toral SL (2019) A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario. Futur Gener Comput Syst 90:129–148
16. Shi W, Li J, Xu W, Zhou H, Zhang N, Zhang S, Shen X (2018) Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks. IEEE Access 6:12518–12529

17. Ghamry KA, Kamel MA, Zhang Y (June 2017) Multiple UAVs in forest fire fighting mission using particle swarm optimization. In: 2017 International conference on unmanned aircraft systems (ICUAS), pp 1404–1409
18. Cheng Z, Wang E, Tang Y, Wang Y (2014) Real-time path planning strategy for UAV based on improved particle swarm optimization. JCP 9(1):209–214
19. Bircher A, Kamel M, Alexis K, Oleynikova H, Siegwart R (May 2016) Receding horizon "next-best-view" planner for 3d exploration. In: 2016 IEEE international conference on robotics and automation (ICRA), pp 1462–1468
20. Teng H, Ahmad I, Msm A, Chang K (2020) 3D optimal surveillance trajectory planning for multiple UAVs by using particle swarm optimization with surveillance area priority. IEEE Access 8:86316–86327
21. Pattanayak S, Choudhury BB (2021) Modified crash-minimization path designing approach for autonomous material handling robot. Evol Intel 14(1):21–34
22. Yu X, Li C, Zhou J (2020) A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios. Knowl-Based Syst 204:106209
23. Dasdemir E, Köksalan M, Öztürk DT (2020) A flexible reference point-based multi-objective evolutionary algorithm: an application to the UAV route planning problem. Comput Oper Res 114:104811
24. Qu C, Gai W, Zhang J, Zhong M (2020) A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (UAV) path planning. Knowl-Based Syst 194, 105530
25. Atencia CR, Del Ser J, Camacho D (2019) Weighted strategies to guide a multi-objective evolutionary algorithm for multi-UAV mission planning. Swarm Evol Comput 44:480–495
26. Shakoor S, Kaleem Z, Do DT, Dobre OA, Jamalipour A (2020) Joint optimization of UAV 3D placement and path loss factor for energy efficient maximal coverage. IEEE Internet Things J 9776–9786
27. Do-Duy T, Nguyen LD, Duong TQ, Khosravirad S, Claussen H (2021) Joint optimisation of real-time deployment and resource allocation for UAV-Aided disaster emergency communications. IEEE J Sel Areas Commun 1–14
28. Nguyen KK, Vien NA, Nguyen LD, Le MT, Hanzo L, Duong TQ (2020) Real-time energy harvesting aided scheduling in UAV-assisted D2D networks relying on deep reinforcement learning. IEEE Access 9:3638–3648
29. Do DT, Nguyen TTT, Le CB, Voznak M, Kaleem Z, Rabie KM (2020) UAV relaying enabled NOMA network with hybrid duplexing and multiple antennas. IEEE Access 8:186993–187007
30. Kaleem Z, Yousaf M, Qamar A, Ahmad A, Duong TQ, Choi W, Jamalipour A (2019) UAV-empowered disaster-resilient edge architecture for delay-sensitive communication. IEEE Network 33(6):124–132
31. Zhou H, Zhang T, Jagadeesan J (2018) Re-weighting and 1-point RANSAC-Based P $ n $ n P solution to handle outliers. IEEE Trans Pattern Anal Mach Intell 41(12):3022–3033
32. Chum O, Matas J (June 2005) Matching with PROSAC-progressive sample consensus. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol 1, pp 220–226
33. Bellavia F, Tegolo D, Valenti C (2011) Improving Harris corner selection strategy. IET Comput Vision 5(2):87–96
34. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 6:679–698
35. Karami E, Prasad S, Shehata M (2017) Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. arXiv:1710.02726
36. Ohta Y, Kanade T (1985) Stereo by intra-and inter-scanline search using dynamic programming. IEEE Trans Pattern Anal Mach Intell 2:139–154
37. Lowe DG (1991) Fitting parameterized three-dimensional models to images. IEEE Trans Pattern Anal Mach Intell 13(5):441–450
38. Zheng C, Li L, Xu F, Sun F, Ding M (2005) Evolutionary route planner for unmanned air vehicles. IEEE Trans Rob 21(4):609–620

39. Kennedy J, Eberhart R (Nov 1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4, pp 1942–1948
40. Roberge V, Tarbouchi M, Labonté G (2012) Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Trans Industr Inf 9(1):132–141
41. Fonder M, Van Droogenbroeck M (2019) Mid-air: a multi-modal dataset for extremely low altitude drone flights. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 0–0
42. Guo X, Chen S, Lin H, Wang H, Wang S (July 2017) A 3D terrain meshing method based on discrete point cloud. In: 2017 IEEE international conference on information and automation (ICIA), pp 12–17
43. Kneip L, Scaramuzza D, Siegwart R (2011) A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. CVPR 2011:2969–2976