JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 43, No. 9, September 2020

Engineering Notes

Generalized Shape Expansion-Based Motion Planning in Three-Dimensional Obstacle-Cluttered Environment

Vrushabh Vijaykumar Zinage* and Satadal Ghosh†
Indian Institute of Technology, Chennai 600 036, India

https://doi.org/10.2514/1.G004756

I. Introduction

THANKS to the advance in communications, control, and actuation, massive improvement in technology of unmanned aerial vehicles (UAVs) has taken place in the last two decades. UAVs also enjoy the advantages of cost-effectiveness, less human risk in operation, convenience of portability, etc. Consequently, unmanned aerial systems (UASs) have seen a steady increase in applications [1]. The applications of UASs range from civilian ones like pollutant monitoring, search and rescue missions, forest fire surveillance, wildlife management, etc., to military ones like security, surveillance and reconnaissance, patrolling border, weapon detection and counter-weapon capabilities, intelligence operations, and so on. An active focus on seamless integration of unmanned systems having diverse unmanned capabilities with manned systems has been highlighted by the U.S. Department of Defense [2].

For satisfactory implementation of a UAS, motion planning of each UAV [also known as (a.k.a.) "agent"] in the UAS is most important. In a generic motion-planning problem, a feasible path (preferably optimal in some sense) is needed to be found for an agent, which has to reach a goal point starting from an initial (start) point while avoiding obstacles in the environment. In general, the obstacles may be both static and dynamic, and the information about the obstacles may be known a priori or unknown. In the literature, motion planners (especially for aerial robots) have widely been discussed over the last three and half decades. One of the seminal papers was on real-time obstacle avoidance by potential field theory [3]. However, this method was shown to suffer from the problem of local minima in Ref. [4]. The guidance community has also presented several online motion-planning strategies with collision avoidance features using several methods like the collision cone approach [5–8], reactive proportional navigation [9], avoidance maps [10], gradient vector fields [11,12], pseudospectral methods [13], the four-parameter logistic curve [14], and the terminal angular spectrum avoidance feature [15]. Motion planning for UAVs has also been reported for an all-aspect approach at a stationary target in Ref. [16], for a GPS-denied environment in Ref. [17], and for evasion from radar tracking in Ref. [18]. Besides these, a methodology for onboard planning and executing trajectories of spacecraft in highly constrained environments was proposed in Ref. [19]. A survey on path planning and path following algorithms, and their comparison, has been presented in Ref. [20].

Another approach for motion planning came into existence with the development of deterministic planning algorithms. Attempt to capture the connectivity of search space led to development of exact search techniques like Voronoi diagrams [21–23], Delaunay triangulation, visibility graphs [24–27], and adaptive roadmaps. Methods like cell decomposition [28], which divided the regions into cells, have also been studied. Search algorithms like Dijkstra's [29], A* [30] were used to find the shortest path in this connectivity graph; whereas D* and AD* algorithms [31] are used in dynamic graphs. Graphical models over a discretized search space were also used for collision-free trajectory planning of UAVs in Refs. [32,33]. The major limitation of these methods is their computational burden, for which their performance degrades severely in higher dimensions and as the size of the environment increases.

Besides these, there has been another methodology of motion planning that follows sampling-based techniques. These techniques avoid the explicit representation of obstacles in the configurations space. For solving high-dimensional and complex motion-planning problems, sampling-based motion-planning algorithms have created a successful algorithmic paradigm shift in motion planning. For an environment (a.k.a. "workspace") known a priori, sampling-based algorithms rely on collision check modules that provide information about the feasibility of candidate paths; and, they connect a set of points sampled from the obstacle-free space to build a graph (a.k.a. "roadmap") of feasible paths, which is then used to construct the solution to the original motion-planning problem [34]. Seminal sampling-based algorithms for robotic motion planning are probabilistic roadmaps (PRM [34] and PRM* [35]), rapidly exploring random trees (RRT [36] and RRT* [35]), and fast marching trees (FMT* [37]). The PRM* algorithm is useful for multiquery problems. A topological graph is constructed called the roadmap, which efficiently solves the multiquery initial/goal states [35]. The major limitation is the heavy collision avoidance checking, which the motion-planning algorithm treats as a "black box" that makes the algorithm computationally expensive. On the other hand, RRT* is useful for single-query problems [35]. In single-query algorithms, a single initial/goal pair is given and the task is to find a feasible solution from initial to goal positions or report failure. These algorithms by default cannot deal with six-degree-of-freedom nonlinear dynamics of UAVs. Extensions of the basic RRT algorithm in several directions have found many applications in the robotics domain [38–42]. The RRTs have also been shown to work effectively with systems involving nonlinear dynamics and differential constraints. Among these algorithms, PRM*, RRT*, and FMT* are asymptotically optimal in the sense that, as the number of samples tends to infinity, the generated shortest path is optimal for distance-based cost functions. These sampling-based methods are computationally efficient because they avoid the explicit representation of obstacles in state space. Another motion-planning algorithm, termed "SE-SCP," which used a spherical expansion method for exploring the workspace, was presented in Ref. [43] and shown to be computationally more efficient than PRM* and RRT*. The word "expansion" refers to the growth of a connected graph of sampled points in the workspace.

To have a further computationally efficient algorithm that sacrifices only very little space in the overall workspace while exploring the same, this Note attempts to expand over the SE-SCP algorithm and presents the generalized shape expansion (GSE) algorithm for a three-dimensional (3-D) environment. Contrary to the shape expansion performed by the SE-SCP, which is restricted to a spherical one with a small radius (equal to the minimum distance from any obstacle), the GSE algorithm helps in expansion over a generalized

Presented as Paper 2020-0860 at the AIAA SciTech Forum 2020, Orlando, Florida, January 6-11, 2020; received 7 August 2019; revision received 18 June 2020; accepted for publication 18 June 2020; published online 22 July 2020.

*Student, Department of Aerospace Engineering; ae16b017@smail.iitm.ac.in.

†Faculty, Department of Aerospace Engineering; satadal@iitm.ac.in.

shape, which is the best representative of the free space in the overall workspace that helps in exploring the free space in a much more efficient way. That is why the word "generalized" is used here. To this end, recently, a sampling-based motion-planning algorithm has been presented in Ref. [44], which explored a two-dimensional (2-D) workspace leveraging the novel GSE algorithm. It was found to explore the free space in a very efficient way, which was reflected in its computational advantage over several other existing seminal algorithms. Considering the advantages of the GSE algorithm and noting that the UAVs are primarily for operation in 3-D environments, this Note presents the 3-D/generalized shape expansion (3D-GSE) algorithm. Note that the major challenge in formulating the 3D-GSE algorithm over the GSE in a planar environment [44] is that in case of the 3D-GSE, the obstacle space in all directions of the entire 3-D environment needs to be avoided, contrary to avoiding obstacle space in all directions on a 2-D plane, as in case of GSE algorithm.

The salient features of the 3D-GSE algorithm are as follows: The method of joining vertices with edges based on the notion of a generalized shape helps the 3D-GSE to avoid computationally heavy collision checking modules. The generalized shape expansion method also has an advantage over spherical expansion in the sense that the 3D-GSE does not restrict the exploration at each step only to a sphere of a small radius that is governed by the distance to the nearest obstacle. Also, since the 3D-GSE algorithm generates a larger generalized shape, it essentially requires less sampled points, which can effectively explore the free space in the environment. Hence, the generated graph has a lower number of vertices; hence, shortest path algorithms like Dijkstra's [29] can compute the shortest path in the graph with a lower number of vertices with more ease. These features render great computational advantage to the presented 3D-GSE algorithm, which is reflected in the comparative simulation studies with respect to (w.r.t.) other algorithms like SE-SCP, PRM*, RRT*, and FMT*. It is also found that the optimal trajectory costs obtained by the 3D-GSE algorithm are slightly better w.r.t. these other algorithms. These indicate a strong potential of the 3D-GSE algorithm to be real-time implementable.

The Note is organized as follows. The problem statement is given in Sec. II. The 3-D generalized shape expansion step of the 3D-GSE algorithm (the key contribution of this Note) is detailed in Sec. III.B, whereas the trajectory optimization step by sequential convex programming is discussed in Sec. III.C. Simulation studies are presented in Sec. IV to demonstrate the performance of the presented 3D-GSE algorithm and its relative performance w.r.t. well-established existing algorithms like PRM*, RRT*, SE-SCP, and FMT*. Finally, concluding remarks are made in Sec. V.

II. Problem Statement

Consider a workspace denoted by $\mathbb{X}$ in a 3-D environment. Hence, $\mathbb{X} \subset \mathbb{R}^3$. The obstacle space is given as $\mathbb{X}_{\text{obs}} \subset \mathbb{X}$. Let $\mathbb{X}_{\text{unsafe}}$ represent the unsafe region around an obstacle that the UAV cannot enter. The unsafe region can be given in the form $\mathbb{X}_{\text{unsafe}} = \mathbb{X}_{\text{obs}} \cup \delta\mathbb{X}_{\text{obs}}$, where $\delta\mathbb{X}_{\text{obs}}$ is the neighborhood of $\mathbb{X}_{\text{obs}}$. For point mass assumption of the UAV, $\mathbb{X}_{\text{unsafe}} = \mathbb{X}_{\text{obs}}$. Otherwise, $\mathbb{X}_{\text{unsafe}}^{\mathbb{X}_{\text{obs}}}$. An entire workspace is considered, as shown in Fig. 1. In this Note, a Cartesian coordinate reference frame is considered. Therefore, the region where the UAV can maneuver freely is given as $\mathbb{X}_{\text{free}} = \mathbb{X} \setminus (\mathbb{X}_{\text{obs}} \cup \mathbb{X}_{\text{unsafe}})$. The initial point $\boldsymbol{X}_{\text{init}} \subset \mathbb{X}_{\text{free}}$ and the final point can be denoted by $\boldsymbol{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$. Boundary points of each obstacle are assumed to be known a priori.

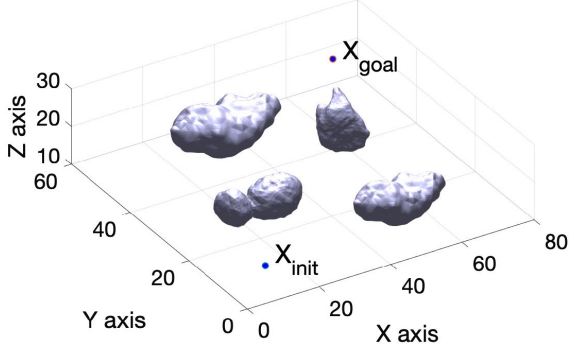


Fig. 1 Sample illustration of workspace.

The UAV's dynamics is given by the following nonlinear equation:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

where f is locally Lipschitz. Here, $\boldsymbol{x}$ is the state vector of the UAV, $\boldsymbol{u} \in \mathbb{U}$ is the control input to the UAV, and $\mathbb{U}$ is the feasible range of its control inputs. Let $\boldsymbol{\gamma}(t) \in \mathbb{X}_{\text{free}}$ represent a feasible trajectory of the UAV and $c(\boldsymbol{\gamma}(t))$ represent the cost incurred by the UAV to track this trajectory. Let the starting position and the goal position of the UAV at times t_o and t_f be given as $\boldsymbol{X}_{\text{init}} \in \mathbb{X}_{\text{free}}$ and $\boldsymbol{X}_{\text{goal}} \in \mathbb{X}_{\text{free}}$, respectively. The optimal motion-planning problem is to design a trajectory $\boldsymbol{\gamma}^*(t) \in \mathbb{X}_{\text{free}}$ from the start position to the goal position so that the cost incurred by the UAV to track this trajectory $c(\boldsymbol{\gamma}(t))$ is minimized. The corresponding mathematical optimization problem can be written as

$$\begin{aligned} &\underset{\boldsymbol{\gamma}(t),\boldsymbol{u}(t)}{\text{minimize}}\ c(\boldsymbol{\gamma}(t)) \\ &\text{subject to } \boldsymbol{\gamma}(t_0) = \boldsymbol{X}_{\text{init}};\ \boldsymbol{\gamma}(t_f) = \boldsymbol{X}_{\text{goal}} \\ &\qquad \dot{\boldsymbol{\gamma}}(t) = f(\boldsymbol{\gamma}(t), \boldsymbol{u}(t)) \quad \forall\ t \in [t_0, t_f) \\ &\qquad \boldsymbol{\gamma}(t) \in \mathbb{X}_{\text{free}};\ \boldsymbol{u}(t) \in \mathbb{U} \quad \forall\ t \in [t_0, t_f) \end{aligned} \tag{2}$$

Obviously, the globally optimal solution of Eq. (2) gives the best possible trajectory for the UAV to follow. However, the computational time burden for exploring the entire workspace and all homotopy classes within it is usually quite high. Instead, a better approach could be to find the best locally optimal solution within the homotopy classes that could be updated relatively fast until the present time step [43]. Considering this, define $\mathbb{X}_{\text{homotopy}} \in \mathbb{X}_{\text{free}}$ as the set of all homotopy classes that have been discovered until the present time step. Therefore, the constraint on $\boldsymbol{\gamma}(t)$ in the amended optimization problem, which would be solved in subsequent sections, is changed to $\boldsymbol{\gamma}(t) \in \mathbb{X}_{\text{homotopy}}\ \forall\ t \in [t_0, t_f)$ instead of $\boldsymbol{\gamma}(t) \in \mathbb{X}_{\text{free}}$, as in Eq. (2).

III. 3D-GSE Algorithm

The 3-D generalized shape expansion algorithm will be described in this section to solve the amended optimization problem stated at the end of Sec. II. Consider a workspace with X_{init}, X_{goal}, and m obstacles as shown in Fig. 1. For a given sampled point $\boldsymbol{X}$, first, the minimum distance of $\boldsymbol{X}$ from the ith obstacle in $\mathbb{X}_{\text{obs}}$, denoted as $r_{\text{i,X}}$, is obtained. Let the point on obstacle i, which is at a minimum distance $r_{\text{i,X}}$ from the sampled point, be given as $\boldsymbol{p}_{\text{i,X}}$. Then, a plane is constructed whose normal is the vector from point $\boldsymbol{X}$ to $\boldsymbol{p}_{\text{i,X}}$ and passing through point $\boldsymbol{p}_{\text{i,X}}$ (for illustration, see Fig. 2). Let this normal vector be denoted by $\boldsymbol{n}_{\text{i,X}}$. Next, the set of points of intersection of this plane and the lines joining the points on obstacle i and the sampled point X are found and denoted as $\mathbb{Q}_{\text{i,X}}$. The maximum distance of point $\boldsymbol{p}_{\text{i,X}}$ from the points in $\mathbb{Q}_{\text{i,X}}$ can be given as $l_{\text{i,X}}$:

$$l_{\text{i,X}} := \underset{\boldsymbol{x} \in \mathbb{Q}_{\text{i,X}}}{\text{argmax}} \|\boldsymbol{p}_{\text{i,X}} - \boldsymbol{x}\|_2 \tag{3}$$

Thus, for obstacle i, the values of four parameters ($r_{\text{i,X}}$, $\boldsymbol{p}_{\text{i,X}}$, $l_{\text{i,X}}$, and $\boldsymbol{n}_{\text{i,X}}$) are computed and stored, where $i = 1, 2, \ldots, m$ and m is the number of obstacles in $\mathbb{X}_{\text{obs}}$. All this information is then used to evaluate a shape classifier function $S_X(\boldsymbol{P})$ for a point $\boldsymbol{P} \in \mathbb{X}$ to check whether the point $\boldsymbol{P}$ belongs to the 3-D generalized shape generated about $\boldsymbol{X}$. This will be elaborated on later. Let $\mathbb{E}$ be the set of edges, $\mathbb{V}$ be the set of vertices, and $G = (\mathbb{V}, \mathbb{E})$ be a directed graph. For each edge in the graph G, the information on start and end points of the edge and the cost associated with the edge are stored.

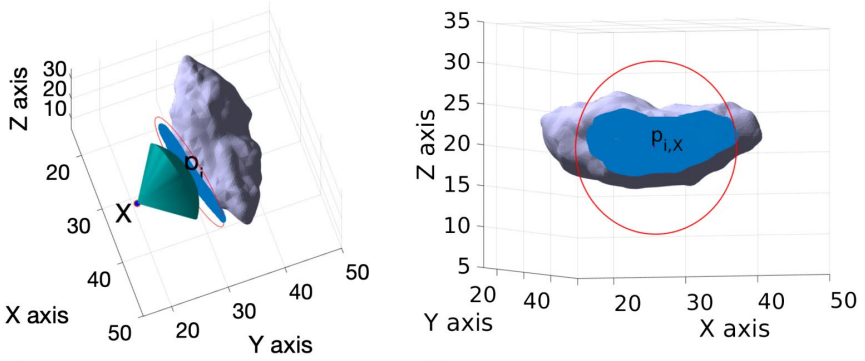


Fig. 2 Spherical sector generated about sample point X: a) side view of spherical sector of radius $r_{i,X}$, and b) front view of spherical sector. Red circle denotes circle of radius $l_{i,X}$ centered at $p_{i,X}$.

A. Initialization Step

In the initialization step, $\boldsymbol{X}_{\text{init}}$ and $\boldsymbol{X}_{\text{goal}}$ are incorporated in $\mathbb{V}$. Another set is defined as $\mathbb{VI}_X = \{r_{\text{i,X}}, \boldsymbol{p}_{\text{i,X}}, l_{\text{i,X}}, \boldsymbol{n}_{\text{i,X}}\}_{i=1}^{m}$ for a point X. These $4m$ parameters in $\mathbb{VI}_X$ for all points $X \in \mathbb{V}$ are available globally to all functions discussed later. Values of these parameters are then used to compute the corresponding shape classifier functions $S_{\text{init}}(\boldsymbol{P})$ and $S_{\text{goal}}(\boldsymbol{P})$, respectively, using the function $\text{Shape}(\boldsymbol{P}, \boldsymbol{X}, \mathbb{X}_{\text{obs}})$ for $\boldsymbol{X} = \boldsymbol{X}_{\text{init}}$ and $\boldsymbol{X}_{\text{goal}}$ (see lines 1 and 2 of Algorithm 1).

The path that connects the initial and goal points by joining vertices of the graph containing X_{init}, X_{goal}, and sampled points in the workspace is referred to as $\text{Path}_{\text{init,goal}}$, whereas $\gamma_{\text{init,goal}}$ denotes a feasible trajectory that an UAV can follow along any such connected path from the initial point to the goal point. Also, note that the cost of a trajectory is given as the length of the trajectory and denoted as $c(\gamma)$. In line 7 of Algorithm 1, the initialization phase cost of a trajectory $c(\gamma_{\text{init,goal}})$ has been set to infinity, which implies that during the initialization stage, the trajectory from the initial point to the goal point is not yet generated.

B. 3D-Generalized Shape Expansion Strategy

1. *Generate Sample Points in* $\mathbb{X}$

In this Note, sample points $\boldsymbol{X}_{\text{rand}}$ are considered to be drawn from a uniform distribution in $\mathbb{X}$ such that the sequence of random points is independent and identically distributed. The function `GenerateSample` in line 9 of Algorithm 1 returns the random variable $\boldsymbol{X}_{\text{rand}} \in \mathbb{X}$.

2. *Generate New Point Using 3D-GSE Algorithm*

For a given $\boldsymbol{X}_{\text{rand}}$, the function `Nearest` in line 10 of Algorithm 1 returns the vertex $\boldsymbol{X}_{\text{nearest}}$ from the set of vertices $\mathbb{V}$ that is nearest to $\boldsymbol{X}_{\text{rand}}$:

$$\texttt{Nearest}(G = (\mathbb{V}, \mathbb{E}), \boldsymbol{X}_{\text{rand}}) \coloneqq \underset{X \in \mathbb{V}}{\arg\min} \|X - \boldsymbol{X}_{\text{rand}}\|_2 \quad (4)$$

Figure 3 shows the steps involved in generation of the 3-D generalized shape for any sampled point $\boldsymbol{X}$. The function `Shape` in lines 1, 2, 11, and 13 is generated as follows: First, indices of m obstacles are assigned in ascending order of $r_{\text{i,X}}$. Thus, without loss of generality,

Algorithm 1: 3D-GSE algorithm

1: $\mathbb{V} \leftarrow \{\boldsymbol{X}_{\text{init}}, \boldsymbol{X}_{\text{goal}}\}$
2: $\mathbb{VI}_{X_{\text{init}}} \leftarrow \{r_{i,X_{\text{init}}}, \boldsymbol{p}_{i,X_{\text{init}}}, l_{i,X_{\text{init}}}, \boldsymbol{n}_{i,X_{\text{init}}}\}_{i=1}^{m}$
3: $\mathbb{VI}_{X_{\text{goal}}} \leftarrow \{r_{i,X_{\text{goal}}}, \boldsymbol{p}_{i,X_{\text{goal}}}, l_{i,X_{\text{goal}}}, \boldsymbol{n}_{i,X_{\text{goal}}}\}_{i=1}^{m}$
4: $S_{\text{init}}(\boldsymbol{P}) \leftarrow \texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}_{\text{init}}, \mathbb{X}_{\text{obs}})$
5: $S_{\text{goal}}(\boldsymbol{P}) \leftarrow \texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}_{\text{goal}}, \mathbb{X}_{\text{obs}})$
6: $\mathbb{E} \leftarrow 0$
7: $c(\text{Path}_{\text{init,goal}}) \leftarrow \infty, c(\gamma_{\text{init,goal}}) \leftarrow \infty$
8: **while** no directed graph is generated from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$
9: $\quad \boldsymbol{X}_{\text{rand}} \leftarrow \texttt{GenerateSample}$
10: $\quad \boldsymbol{X}_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (\mathbb{V}, \mathbb{E}), \boldsymbol{X}_{\text{rand}})$
11: $\quad S_{\text{nearest}}(\boldsymbol{P}) \leftarrow \texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}_{\text{nearest}}, \mathbb{X}_{\text{obs}})$
12: $\quad \boldsymbol{X}_{\text{new}} \leftarrow \texttt{Steer}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{rand}})$
13: $\quad S_{\text{new}}(\boldsymbol{P}) \leftarrow \texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}_{\text{new}}, \mathbb{X}_{\text{obs}})$
14: $\quad \boldsymbol{X}_{\text{near}} \leftarrow \texttt{NearIntersectedShapes}(G = (\mathbb{V}, \mathbb{E}), \boldsymbol{X}_{\text{new}})$
15: $\quad \mathbb{V} \leftarrow \mathbb{V} \cup \boldsymbol{X}_{\text{new}}$
16: $\quad \mathbb{VI}_{X_{\text{new}}} \leftarrow \{r_{i,X_{\text{new}}}, \boldsymbol{p}_{i,X_{\text{new}}}, l_{i,X_{\text{new}}}, \boldsymbol{n}_{i,X_{\text{new}}}\}_{i=1}^{m}$
17: $\quad$ **for** $\boldsymbol{X}_{\text{n}} \in \boldsymbol{X}_{\text{near}}$ **do**
18: $\quad\quad \text{Path}_{\text{n,new}}, c(\text{Path}_{\text{n,new}}), \text{Path}_{\text{new,n}}, c(\text{Path}_{\text{new,n}}) \leftarrow \texttt{Path}(\boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{new}})$
19: $\quad\quad \mathbb{E} \leftarrow \mathbb{E} \cup \{(\boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{new}})[\text{Path}_{\text{n,new}}, c(\text{Path}_{\text{new,n}})], (\boldsymbol{X}_{\text{new}}, \boldsymbol{X}_{\text{n}})[\text{Path}_{\text{n,new}}, c(\text{Path}_{\text{new,n}})]\}$
20: $\quad$ **end for**
21: **end while**
22: $\{\text{Path}_{\text{init,goal,shortest}}, c(\text{Path}_{\text{init,goal,shortest}})\} \leftarrow \texttt{MinPath}(G = (\mathbb{V}, \mathbb{E}), \boldsymbol{X}_{\text{init}}, \boldsymbol{X}_{\text{goal}})$
23: $\{\gamma_{\text{init,goal,shortest}}, c(\gamma_{\text{init,goal,shortest}})\} \leftarrow \texttt{SCPOptimalTrajectory}(\boldsymbol{X}_{\text{init}}, \boldsymbol{X}_{\text{goal}}, \{\text{Path}_{\text{init,goal,shortest}}\})$
24: $\gamma_{\text{init,goal,optimal}}, c(\gamma_{\text{init,goal,optimal}}) \leftarrow \texttt{MinCostTrajectory}(\{\gamma_{\text{init,goal,shortest}}, c(\gamma_{\text{init,goal,shortest}})\})$

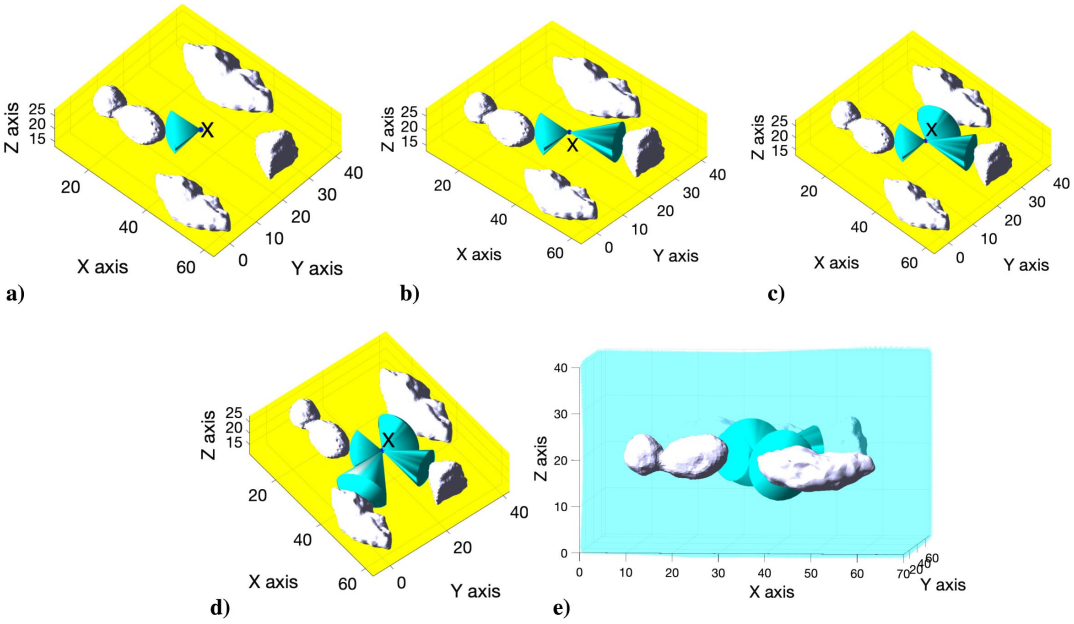


Fig. 3 Steps involved in generation of 3-D generalized shape for a sample point X.

let $r_{1,X} \leq r_{2,X} \leq \ldots \leq r_{m,X}$. Then, for the ith obstacle for $i \in \{1, 2, \ldots, m\}$, $r_{i,X}$, $\boldsymbol{p}_{i,X}$, $l_{i,X}$, and $\boldsymbol{n}_{i,X}$ values are retrieved.

Now, consider the example with a workspace with $m = 4$ obstacles as shown in Fig. 1. Consider a sampled point $\boldsymbol{X}$ about which the 3-D generalized shape is to be generated. First, a spherical sector (say, $\mathcal{R}_{1,X}$ as shown in Fig. 3a) is created with the vertex at the sampled point $\boldsymbol{X}$ and the radius equal to $r_{1,X}$, the cone angle equal to $\tan^{-1}(l_{1,X}/r_{1,X})$, and the axis along $\boldsymbol{n}_{1,X}$. After this, another spherical sector (say, $\mathcal{R}_{2,X}$) is generated with the vertex at the sampled point $\boldsymbol{X}$ radius equal to $r_{2,X}$, the cone angle equal to $\tan^{-1}(l_{2,X}/r_{2,X})$, and the axis along $\boldsymbol{n}_{2,X}$. Similarly, $\mathcal{R}_{3,X}$ and $\mathcal{R}_{4,X}$ are generated. The 3-D generalized shape $\mathcal{S}_X$ is defined as follows and shown in Fig. 3e (shaded region):

$$\mathcal{S}_X = \mathcal{R}_{1,X} \cup (\mathcal{R}_{2,X} \setminus \mathcal{R}_{1,X}) \cup (\mathcal{R}_{3,X} \setminus (\mathcal{R}_{2,X} \cup \mathcal{R}_{1,X})) \cup (\mathcal{R}_{4,X} \setminus (\mathcal{R}_{1,X} \cup \mathcal{R}_{2,X} \cup \mathcal{R}_{3,X})) \cup \mathcal{I} \tag{5}$$

Here, The difference $(\mathcal{R}_{2,X} \setminus \mathcal{R}_{1,X})$ denotes set-theoretic subtraction of spherical sector $\mathcal{R}_{1,X}$ from $\mathcal{R}_{2,X}$, which can be mathematically given as

$$\mathcal{R}_{2,X} \setminus \mathcal{R}_{1,X} = \{x : x \in \mathcal{R}_{2,X} \text{ and } x \notin \mathcal{R}_{1,X}\}$$

All other terms, except the last term in the right-hand side of Eq. (7), are also obtained as such set-theoretic differences. The last term there, $\mathcal{I}$, is a set defined as

$$\mathcal{I} \subset \mathbb{X} | \mathcal{I} \cap \{\cup_{i=1}^{i=4} \widehat{\mathcal{R}_{i,X}}\} = \varnothing$$

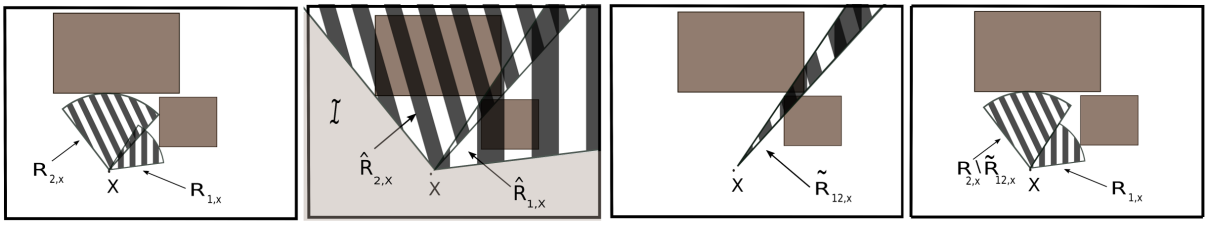
where $\widehat{\mathcal{R}_{i,X}}$ is the intersection of $\mathbb{X}$ and the spherical sector with the vertex at sampled point $\boldsymbol{X}$, cone angle $\tan^{-1}(l_{i,X}/r_{i,X})$, the axis along $\boldsymbol{n}_{i,X}$, and the radius as infinite. An illustration of $\widehat{\mathcal{R}_{i,X}}$ and $\mathcal{I}$ in a 2-D environment is shown in Fig. 4b. Thus, the region $\widehat{\mathcal{R}_{i,X}}$ is essentially obtained by extending the spherical sector $\mathcal{R}_{i,X}$ to the boundary of $\mathbb{X}$ along $\boldsymbol{n}_{i,X}$. The set $\mathcal{I}$ denotes the remaining portion of the free space seen from "X" after considering all spherical sectors. By the definition of the generalized shape and the representation of it in Fig. 3e, this remaining portion of free space is also part of the generalized shape. Thus, $\mathcal{I}$ is defined as a complement of $\{\cup_{i=1}^{i=m} \widehat{\mathcal{R}_{i,X}}\}$. Figure 3e is an illustration of the 3-D generalized shape about sampled point $\boldsymbol{X}$. However, for a generic case of m obstacles, another region $\widetilde{\mathcal{R}_{ij,X}}$ (where $i < j$) is defined as

$$\widetilde{\mathcal{R}_{ij,X}} \triangleq \widehat{\mathcal{R}_{i,X}} \cap \widehat{\mathcal{R}_{j,X}} \tag{6}$$

Thus, the region $\widetilde{\mathcal{R}_{ij,X}}$ (where $i < j$) is the intersection of extensions of regions $\mathcal{R}_{i,X}$ and $\mathcal{R}_{j,X}$, which is intersection of $\widehat{\mathcal{R}_{i,X}}$ and $\widehat{\mathcal{R}_{j,X}}$ as shown in Fig. 4c as an illustration. Note that if $\mathcal{R}_{i,X} \cap \mathcal{R}_{j,X} = \varnothing$, then $\widetilde{\mathcal{R}_{ij,X}} = \varnothing$. It can easily be shown that for any $\mathcal{R}_{i,X}$, $\mathcal{R}_{j,X}$, and $\mathcal{R}_{k,X}$ where $i < j < k$, $(\mathcal{R}_{i,X} \cup \widetilde{\mathcal{R}_{j,X}})_{k,X} = \widetilde{\mathcal{R}_{ik,X}} \cup \widetilde{\mathcal{R}_{jk,X}}$. This can similarly be extended for more number of regions. Therefore, for m number of obstacles in a generic obstacle space, the 3-D generalized shape about $\boldsymbol{X}$ is given as

$$\mathcal{S}_X = \mathcal{R}_{1,X} \cup (\mathcal{R}_{2,X} \setminus \widetilde{\mathcal{R}_{12,X}}) \cup (\mathcal{R}_{3,X} \setminus (\widetilde{\mathcal{R}_{23,X}} \cup \widetilde{\mathcal{R}_{13,X}})) \cup \ldots \cup (\mathcal{R}_{m,X} \setminus \cup_{j=1}^{j=m-1} \widetilde{\mathcal{R}_{jm,X}}) \cup \mathcal{I} \tag{7}$$

where the set $\mathcal{I}$ can be given as



a) $\mathcal{R}_{1,X}$ and $\mathcal{R}_{2,X}$ **b) $\widehat{\mathcal{R}_{1,X}}$, $\widehat{\mathcal{R}_{2,X}}$ and $\mathcal{I}$** **c) $\widetilde{\mathcal{R}_{12,X}}$** **d) $\mathcal{R}_{2,X} \setminus \widetilde{\mathcal{R}_{12,X}}$**

Fig. 4 Illustration of spherical sectors and related regions in 2-D environment.

Algorithm 2: Shape classifier function ($\texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}, \mathbb{X}_{\text{obs}})$)

$r_0 = 0$
for $i = 1, \ldots, m$ **do**
  $\theta_i \leftarrow \tan^{-1}\left(\frac{l_{i,X}}{r_{i,X}}\right)$
  $r_i \leftarrow \text{sat}\left(\cos^{-1}\left(\frac{\boldsymbol{n}_{i,X}\cdot(\boldsymbol{P}-\boldsymbol{X})}{\|\boldsymbol{n}_{i,X}\|_2\|\boldsymbol{P}-\boldsymbol{X}\|_2}\right) - \theta_i\right)$
  $f_i \leftarrow \text{sat}\left(\theta_i - \cos^{-1}\left(\frac{\boldsymbol{n}_{i,X}\cdot(\boldsymbol{P}-\boldsymbol{X})}{\|\boldsymbol{n}_{i,X}\|_2\|\boldsymbol{P}-\boldsymbol{X}\|_2}\right)\right) + \text{sat}(r_{i,X} - \|\boldsymbol{P}-\boldsymbol{X}\|_2)$
end for
if $\text{all}(r_i) = 1$ for $i \in \{1 \ldots, m\}$, **then**
  $\texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}, \mathbb{X}_{\text{obs}}) = 0$
else
  $g_i \leftarrow f_i + \sum_{j=1}^{i=1} r_j - (i+1)$
  $\texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}, \mathbb{X}_{\text{obs}}) \leftarrow g_1 * g_2 * g_3 *, \ldots, g_m$
end if

$$\mathcal{I} \subset \mathbb{X} | \mathcal{I} \cap \{\cup_{i=1}^{i=m} \widehat{\mathcal{R}_{i,X}}\} = \varnothing$$

where $\widehat{\mathcal{R}_{i,X}}$ is as defined earlier. Thus, the following steps are involved in obtaining $\mathcal{S}_X$. First, the spherical sector corresponding to the smallest $r_{i,X}$ (that is, $r_{1,X}$) is included in $\mathcal{S}_X$. Then, the subset of the spherical sector $R_{2,X}$ that lies outside $\mathbb{X}_{\text{unsafe}}$ (that is, $\mathcal{R}_{2,X} \setminus \widetilde{\mathcal{R}_{12,X}}$) is included in $\mathcal{S}_X$. Subsequently, the subset of other spherical sectors, which are outside the unsafe region $\mathbb{X}_{\text{unsafe}}$, are included in $\mathcal{S}_X$. These steps are illustrated in a 2-D environment with two obstacles in Fig. 4.

Note that the generalized shape cannot be defined just as the union of all the spherical sectors because this union could also contain unsafe regions as well, as can be seen in Fig. 4a as an example; whereas the shape should not contain any unsafe region. Now, the pseudocode for the classifier function to classify whether point $\boldsymbol{P}$ is inside the 3-D generalized shape about a sampled point $\boldsymbol{X}$ for m obstacles (that is, whether $\boldsymbol{P} \in S_X$) is given as follows. Here, the function $\text{sat}(x)$ returns a value of one if $x \geq 0$ and zero otherwise. The function $\texttt{Shape}\ (\boldsymbol{P}, \boldsymbol{X}, \mathbb{X}_{\text{obs}})$ in Algorithm 2 returns a value of zero if point $\boldsymbol{P}$ is within the 3-D generalized shape about the sampled point $\boldsymbol{X}$, and it is nonzero otherwise; that is,

$$S_X(\boldsymbol{P}) \begin{cases} = 0; & \text{if } \boldsymbol{P} \in \mathcal{S}_X \\ \neq 0, & \text{otherwise} \end{cases} \tag{8}$$

The functions f_i and r_i, which are mentioned in Algorithm 2 to obtain the shape classifier function $S_X(\boldsymbol{P})$, are elaborated on in the following, with illustrations for the three cases as shown in Fig. 5. Consider any general point $\boldsymbol{P}$ as shown in Fig. 5. Three cases may happen.

1) If the point is inside $\widehat{\mathcal{R}_{i,X}}$ but outside the spherical sector $\mathcal{R}_{i,X}$, as shown in Fig. 5a, then in this case, $f_i = 1$ and $r_i = 0$. Therefore, $f_i \neq 2$ and $r_i = 0$.

2) If the point is inside the spherical sector $\mathcal{R}_{i,X}$, as shown in Fig. 5b, then $f_i = 2$ and $r_i = 0$.

3) If the point is outside $\widehat{\mathcal{R}_{i,X}}$ as well as the spherical sector $\mathcal{R}_{i,X}$ as shown in Fig. 5c, then $f_i \neq 2$ and $r_i = 1$.

Thus, for a generic case of m obstacles, the function f_i is directly related to $\mathcal{R}_{i,X}$ only as described earlier in this paper; that is, $f_i = 2$ if and only if $\boldsymbol{P} \in \mathcal{R}_{i,X}$. And, $r_i = 1$ if and only if $\boldsymbol{P}$ does not belong to $\widehat{\mathcal{R}_{i,X}}$ such that $g_i = f_i + r_{i-1} + r_{i-2} + \ldots r_1 - (i+1)$ (see Algorithm 2) is related to $\mathcal{R}_{i,X} - (\widetilde{\mathcal{R}_{1i,X}} + \ldots \widetilde{\mathcal{R}_{(i-1)i,X}})$ as $g_i = 0$ if and only if point $\boldsymbol{P}$ belongs to the set $\mathcal{R}_{i,X} - (\widetilde{\mathcal{R}_{1i,X}} + \ldots \widetilde{\mathcal{R}_{(i-1)i,X}})$, and it is $g_i < 0$ otherwise. The condition $\text{all}(r_i) = 1$ denotes that point $\boldsymbol{P}$ lies in set $\mathcal{I}$, which was defined earlier as $\mathcal{I} \subset \mathbb{X} | \mathcal{I} \cap \{\cup_{i=1}^{i=m} \widehat{\mathcal{R}_{i,X}}\} = \varnothing$. Finally, these formulations of f_i, r_i, g_i, and $\mathcal{I}$ lead to the formulation of the 3-D generalized shape classifier function $\texttt{Shape}(\boldsymbol{P}, \boldsymbol{X}, \mathbb{X}_{\text{obs}})$, which is zero if and only if point $\boldsymbol{P}$ belongs to the 3-D generalized shape about $\boldsymbol{X}$; that is, $\boldsymbol{P} \in \mathcal{S}_X$, where $\mathcal{S}_X$ is as given in Eq. (7). For example, in Figs. 5a and 5c, $g_i < 0$ implied that point P does not belong to the 3-D generalized shape about sampled point X, whereas in Fig. 5b, $g_i = 0$ implied $\boldsymbol{P}$ belongs to S_X.

Remark: Figures 6a and 6b show the cross section of the entire environment at a specific value of z. Figure 6a depicts the generalized shape that is generated by the GSE algorithm when run on this cross-section plane. Figure 6b depicts the cross section of the generalized shape generated by the 3D-GSE over the entire 3-D environment. Note that since the 3D-GSE considers avoiding the obstacle space in all directions in the 3-D environment, the cross section of its

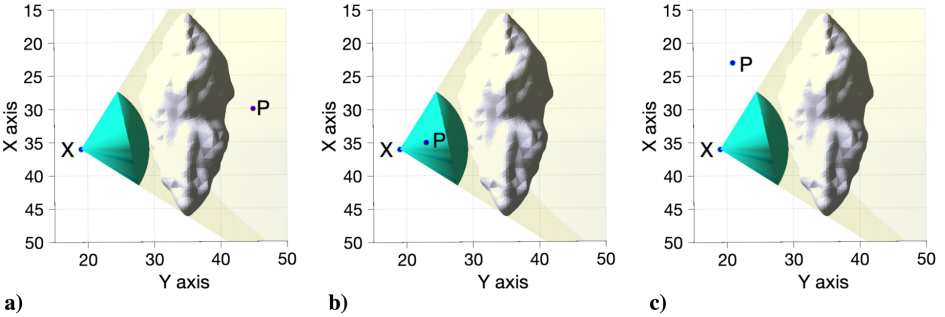


Fig. 5 Computation on f_i and r_i.

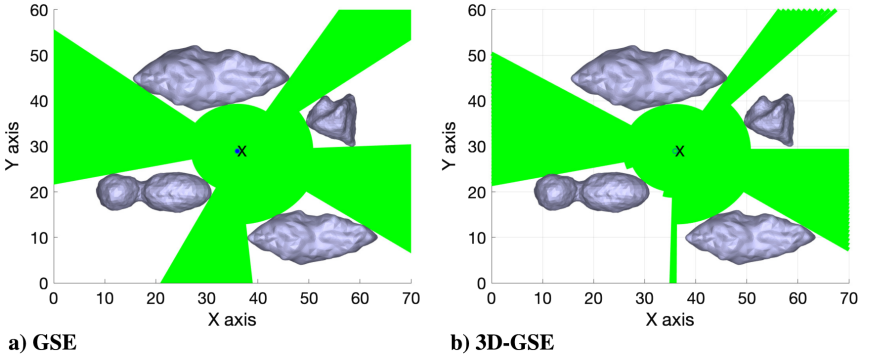


Fig. 6 Cross section of generalized shapes at a particular z value of GSE and 3D-GSE algorithms.

Algorithm 3: Steer function ($\texttt{Steer}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{rand}})$)

1: **if** $S_{\text{nearest}}(\boldsymbol{X}_{\text{rand}}) = 0$, **then**
2: $\quad \boldsymbol{X}_{\text{new}} \leftarrow \boldsymbol{X}_{\text{rand}}$
3: $\quad$ **return** $\boldsymbol{X}_{\text{new}}$
4: **else**
5: $\quad$ **for** $i = m, \ldots, 1$ **do**
6: $\quad\quad \boldsymbol{X}_{\text{new}} \leftarrow \texttt{onShape}(\boldsymbol{X}_{\text{rand}}, \boldsymbol{X}_{\text{nearest}}, r_{\text{i},X_{\text{nearest}}})$
7: $\quad\quad$ **if** $S_{\text{nearest}}(\boldsymbol{X}_{\text{new}}) = 0$, **then**
8: $\quad\quad\quad$ **return** $\boldsymbol{X}_{\text{new}}$
9: $\quad\quad$ **else**
10: $\quad\quad\quad i \leftarrow i - 1$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end if**

generated shape (Fig. 6b) is a subset of the shape generated by GSE on the same plane of cross section of the 3-D environment.

3. Steering $\boldsymbol{X}_{\text{rand}}$ to $\boldsymbol{X}_{\text{new}}$ on 3-D Generalized Shape

Once the Shape function $S_X(\boldsymbol{P})$ is computed for $\boldsymbol{X} = \boldsymbol{X}_{\text{nearest}}$ in line 11 of Algorithm 1, the steer function is computed as mentioned in line 12 of Algorithm 1 to steer the randomly selected point $\boldsymbol{X}_{\text{rand}}$ to a suitable $\boldsymbol{X}_{\text{new}}$. In the steering process, if the point $\boldsymbol{X}_{\text{rand}}$ belongs to the 3-D generalized shape generated by $\boldsymbol{X}_{\text{nearest}}$, then the function returns the point $\boldsymbol{X}_{\text{new}} = \boldsymbol{X}_{\text{rand}}$. Otherwise, the steer function returns the point $\boldsymbol{X}_{\text{new}}$ that is on the boundary of the generalized shape about $\boldsymbol{X}_{\text{nearest}}$ and closest to $\boldsymbol{X}_{\text{rand}}$. To achieve this, the function onShape $(\boldsymbol{X}_{\text{rand}}, \boldsymbol{X}_{\text{nearest}}, r_{i,X_{\text{nearest}}})$ in line 6 of Algorithm 3 returns a point lying on line joining $\boldsymbol{X}_{\text{nearest}}$ and $\boldsymbol{X}_{\text{rand}}$ at a distance of

$$\max_{i \in 1,\ldots,m} \{r_{i,X_{\text{nearest}}}\}$$

from point $\boldsymbol{X}_{\text{nearest}}$. Note that this newly obtained point $\boldsymbol{X}_{\text{new}}$ is the point that is farthest from $\boldsymbol{X}_{\text{nearest}}$ along the line $\boldsymbol{X}_{\text{nearest}} - \boldsymbol{X}_{\text{rand}}$ in the 3-D generalized shape about $\boldsymbol{X}_{\text{nearest}}$. Note that in the spherical expansion algorithm presented in Ref. [43], only $r_{1,X_{\text{nearest}}}$ has been used for this steering function, which enforced $\boldsymbol{X}_{\text{new}}$ to be on a restricted sphere with its radius as

$$\min_{i \in 1,\ldots,m} \{r_{i,X_{\text{nearest}}}\}$$

from point $\boldsymbol{X}_{\text{nearest}}$, whereas the 3D-GSE algorithm relaxes this restriction to the greatest possible extent, which essentially leads to much faster running time, as can be seen in the simulation results presented in Sec. IV.

4. Generation of New Edges

The function

$$\texttt{NearIntersectedShapes}\,(G = (\mathbb{V}, \mathbb{E}), \boldsymbol{X}_{\text{new}})$$

on line 14 in Algorithm 1 generates a set $\mathbb{X}_{\text{near}}$ of all the vertices whose 3-D generalized shapes intersect with the 3-D generalized shape generated about point $\boldsymbol{X}_{\text{new}}$. The pseudocode for NearIntersectedShapes is given in Algorithm 4.

Algorithm 4 NearIntersectedShapes ($\texttt{NearIntersectedShapes}\,(\boldsymbol{G = (\mathbb{V}, \mathbb{E}), X_{\text{new}}})$)

1: for $\boldsymbol{X}_{\text{n}} \in \mathbb{V}$, **do**
2: $\quad$ **if** $\texttt{Shape}(\texttt{Steer}(\boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{new}}), \boldsymbol{X}_{\text{new}}, \mathbb{X}_{\text{obs}}) = 0$
3: $\quad\quad \mathbb{E} \leftarrow \mathbb{E} \cup \{(\boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{new}})[\text{Path}_{\text{n,new}}, c(\text{Path}_{\text{new,n}})], (\boldsymbol{X}_{\text{new}}, \boldsymbol{X}_{\text{n}})[\text{Path}_{\text{n,new}}, c(\text{Path}_{\text{new,n}})]\}$
4: $\quad\quad \mathbb{X}_{\text{near}} \leftarrow \mathbb{X}_{\text{near}} \cup \boldsymbol{X}_{\text{n}}$
5: $\quad$ **end if**
6: **end for**

$$\mathbb{X}_{\text{near}} \coloneqq \{\boldsymbol{X}_{\text{n}} \in \mathbb{V}: \texttt{Shape}(\texttt{Steer}(\boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{new}}), \boldsymbol{X}_{\text{new}}, \mathbb{X}_{\text{obs}}) = \boldsymbol{0}\} \tag{9}$$

The edges of the graph that connect the new vertex $\boldsymbol{X}_{\text{new}}$ with the existing vertices in $\boldsymbol{X}_{\text{near}}$ are now generated. For each vertex $\boldsymbol{X}_{\text{n}} \in \mathbb{X}_{\text{near}}$, the function Path $(\boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{new}})$, mentioned in line 18 of Algorithm 1, is used to generate the two paths between $\boldsymbol{X}_{\text{n}}$ and $\boldsymbol{X}_{\text{new}}$. The first path from $\boldsymbol{X}_{\text{n}}$ to $\boldsymbol{X}_{\text{new}}$ is given by $\boldsymbol{X}_{\text{n,new}}$, and its cost is given by $c(\text{Path}_{\text{n,new}})$. Similarly, the second path from $\boldsymbol{X}_{\text{new}}$ to $\boldsymbol{X}_{\text{n}}$ is given by $\text{Path}_{\text{new,n}}$ and costs $c(\text{Path}_{\text{new,n}})$. These edges are added to the set of edges $\mathbb{E}$, where the corresponding path and their costs are also stored. A sample illustration of the iterations for growth of this graph are shown in Fig. 7.

C. Locally Optimal Trajectory Generation Step

During this step, the shortest path(s) in the graph $G = (\mathbb{V}, \mathbb{E})$ from the initial position $\boldsymbol{X}_{\text{init}}$ to the goal position $\boldsymbol{X}_{\text{goal}}$ is (are) found, and then the locally optimal trajectory is found using sequential convex programming (SCP).

1. Generate Shortest Path

Once the directed graph $G = (\mathbb{V}, \mathbb{E})$ is generated such that $\boldsymbol{X}_{\text{init}}$ and $\boldsymbol{X}_{\text{goal}}$ are connected by paths consisting of all possible combinations of the vertices in $\mathbb{V}$, then the function $\texttt{MinPath}(G = (\mathbb{V}, \mathbb{E}), \boldsymbol{X}_{\text{init}}, \boldsymbol{X}_{\text{goal}})$ in line 22 of Algorithm 1 finds the minimum cost path(s) in the graph $G = (\mathbb{V}, \mathbb{E})$ from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$. Graph search algorithms like Dijkstra's algorithm [29] have been used in this Note for this step. The function MinPath outputs the set of shortest paths $\{\text{Path}_{\text{init,goal,shortest}}\}$ and the cost for traversing these paths $\{c(\text{Path}_{\text{init,goal,shortest}})\}$. Note that the set of shortest paths $\{\text{Path}_{\text{init,goal,shortest}}\}$ may be a singleton or may have more than one path.

2. Sequential Convex Programming Step

A locally optimal trajectory about each of the shortest paths in the set $\{\text{Path}_{\text{init,goal,shortest}}\}$ is obtained using the function

$$\texttt{SCPOptimalTrajectory}(\boldsymbol{X}_{\text{init}}, \boldsymbol{X}_{\text{goal}}, \text{Path}_{\text{init,goal,shortest}})$$

in Line 23 of Algorithm 1. Among these trajectories, the one with minimal cost is the optimal trajectory $\gamma_{\text{init,goal,optimal}}$ found using the function MinCostTrajectory in line 24 of Algorithm 1, and the corresponding path $\text{Path}_{\text{init,goal,optimal}}$ is the optimal path from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$.

In this step, time $[t_0, t_{\text{f}}]$ is decomposed into multiple discrete steps $\{0, \ldots, T\}$, and the amended form of continuous time optimization problem in Eq. (2) (amendment mentioned at the end of Sec. II) is represented as a discrete optimization problem. Then, the function SCPOptimalTrajectory uses SCP to compute the optimal trajectory from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$ that passes through the 3-D generalized shapes corresponding to the vertices:

$$\{\boldsymbol{X_0}, \boldsymbol{X_1}, \ldots, \boldsymbol{X}_{\text{T}-1}\} \in \text{Path}_{\text{init,goal,shortest}}$$

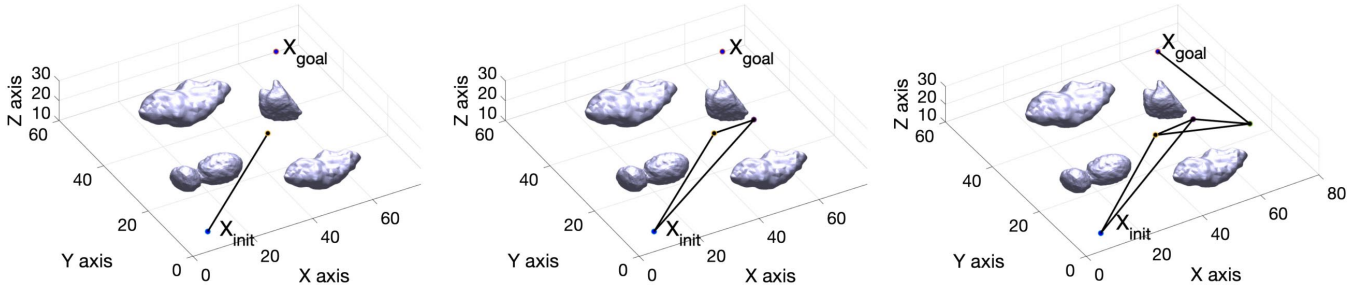
as

$$\underset{\boldsymbol{\gamma}(n), \boldsymbol{u}(n)}{\text{minimize}} \|\boldsymbol{u}(\boldsymbol{n})\|_2$$

subject to

$$\begin{aligned}
&\boldsymbol{\gamma}(0) = \boldsymbol{X}_{\text{init}};\ \boldsymbol{\gamma}(T) = \boldsymbol{X}_{\text{goal}} \\
&\boldsymbol{\gamma}(n+1) = A(n)\boldsymbol{\gamma}(n) + B(n)\boldsymbol{u}(\boldsymbol{n}) \quad \forall\ n \in 0, \ldots, T-1 \\
&\texttt{Shape}(\boldsymbol{\gamma}(n), \boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{obs}}) = 0;\ \|\boldsymbol{u}(\boldsymbol{n})\|_2 \le U_{\max} \quad \forall\ n \in 0, \ldots T-1
\end{aligned} \tag{10}$$

where matrices $A(n)$ and $B(n)$ represent UAV dynamics. Note that $\texttt{Shape}(\boldsymbol{\gamma}(n), \boldsymbol{X}_{\text{n}}, \boldsymbol{X}_{\text{obs}}) = 0\ \forall\ n \in 0, \ldots, T-1$ is equivalent to the



a) Iteration 1 b) Iteration 2 c) Iteration 3

Fig. 7 Multiple iterations of 3D-GSE step.

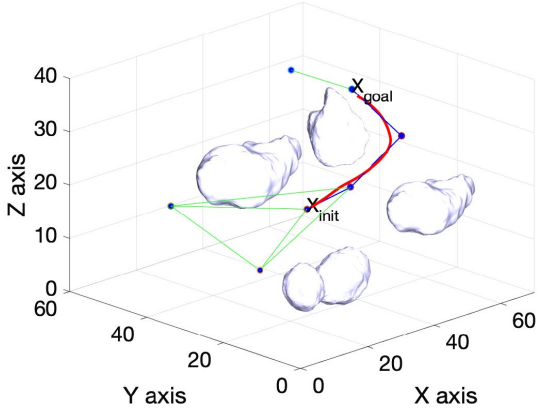


Fig. 8 Generating shortest path on the graph and trajectory optimization over this path.

constraint $\gamma(t) \in \mathbb{X}_{\text{homotopy}}$ mentioned at the end of Sec. II. An illustration of this step of the generation of a locally optimal trajectory about a connected path is shown in Fig. 8.

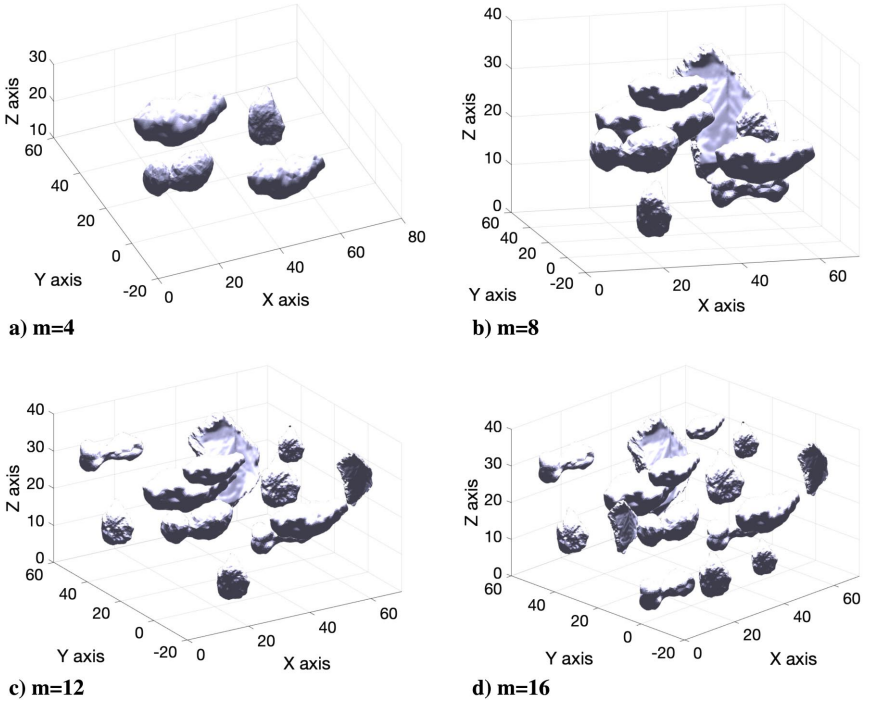
IV. Numerical Simulations

In this section, simulation results are presented for motion planning in workspaces cluttered with $m = 4, 8, 12$, and 16 obstacles, as shown in Fig. 9, representing low to high obstacle density environments, respectively. In all the cases, obstacle spaces are represented as mesh models. As seen in Fig. 9, obstacles can be generic in nature, and no particular assumption about the obstacles is made. The simulation studies have been carried out using MATLAB R2017b on an Intel Core i7 2.2 GHz processor. Besides the simulation studies for the 3D-GSE algorithm presented in this Note, simulation studies have also been performed for RRT* [35], PRM* [35], FMT* [37] and SE-SCP [43] for the sake of comparison. For that purpose, for each of the values of m, four different randomly generated obstacle spaces have been considered and, for each of these obstacle spaces, 100 simulations have been run for each algorithm under comparison.

Figure 10 shows the feasible shortest path from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$ for a given obstacle space containing four obstacles from the constructed graphs using 3D-GSE, RRT*, PRM*, FMT*, and SE-SCP algorithms (Figs. 10a–10e, respectively). The bold line shown in red color for each algorithm represents the shortest path from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$ generated by the algorithm.

Average running times required by the aforementioned motion-planning algorithms to generate the feasible shortest path from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$ are depicted in Fig. 11. The bars in the histogram plots in Fig. 11 represent the average running times (averaged over 100 runs) for each of the four randomly generated obstacle spaces with the number of obstacles of $m = 4, 8, 12$, and 16, which are shown in Figs. 11a–11d, respectively.

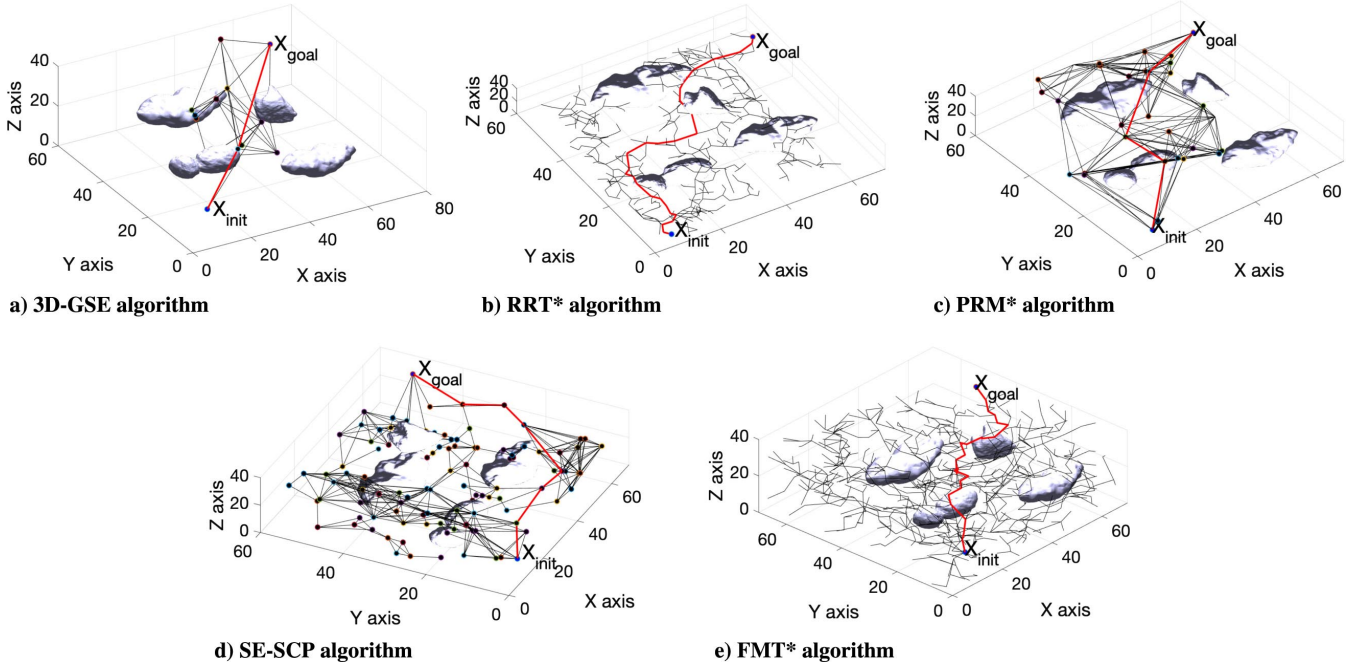
Similarly, average trajectory costs of optimal trajectories from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{goal}}$ obtained by the aforementioned motion-planning algorithms



a) m=4 b) m=8

c) m=12 d) m=16

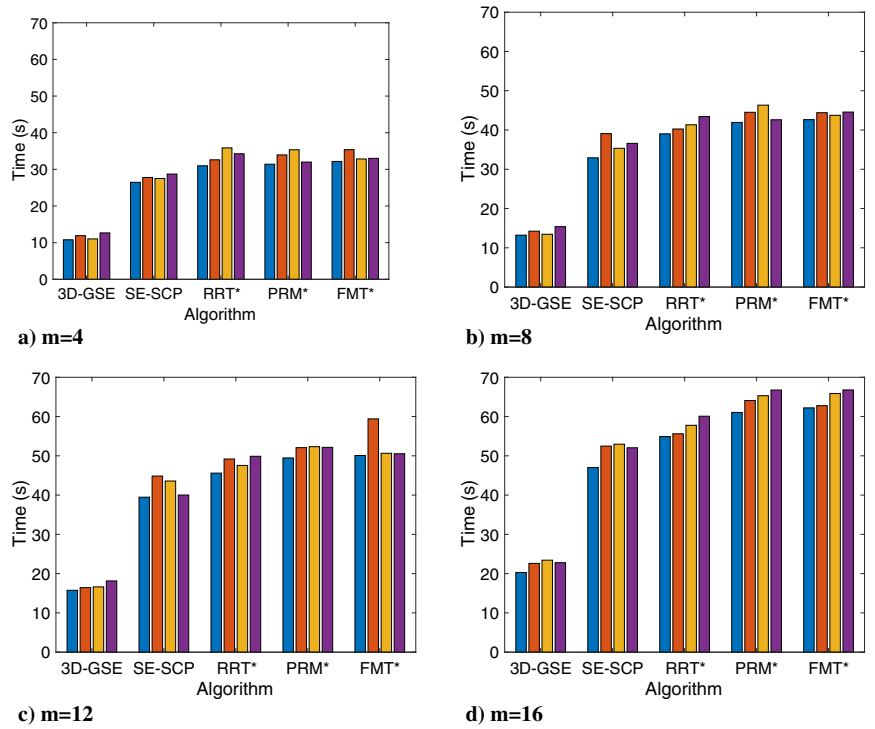
Fig. 9 Obstacle workspace with different numbers of obstacles.



a) 3D-GSE algorithm b) RRT* algorithm c) PRM* algorithm

d) SE-SCP algorithm e) FMT* algorithm

Fig. 10 Shortest path using different algorithms.



a) m=4 b) m=8

c) m=12 d) m=16

Fig. 11 Average running time with various algorithms.

are depicted in Fig. 12. The bars in the histogram plots in Fig. 12 represent the average optimal trajectory costs (averaged over 100 runs) for each of the four randomly generated obstacle spaces, with the number of obstacles of $m = 4, 8, 12$, and 16 shown in Figs. 12a–12d, respectively.

From Fig. 11, it is evident that the 3D-GSE algorithm presented in this Note outperforms other algorithms under comparison in terms of the average running time. This is mainly because of two main reasons. First, the 3-D generalized shape is such generated that the collisions with obstacles are avoided instead of executing a separate computationally heavy collision check algorithm. Note that this advantage was also enjoyed by the SE-SCP algorithm. However, the second reason is that contrary to the spherical shape expansion performed by the SE-SCP, which is restricted to a small radius (equal to the minimum distance from any obstacle), the 3D-GSE expands the shape to a 3-D generalized shape, which is the best representative of the free space in the workspace. Thus, the 3D-GSE can generate the shortest path even more quickly than the SE-SCP and other algorithms. The significant difference in average running time is also evident from the box plots for all runs for all obstacle space for each of $m = 4, 8, 12$, and 16 in Figs. 13a–13d, respectively.

Also, note that the 3D-GSE has no predefined step size, and hence is easily adaptable to different obstacle densities. This can also be noticed from the trends in average running time in Figs. 11 and 13. This feature along with the notion of the 3-D generalized shape expansion, this algorithm can build larger shapes, and hence can

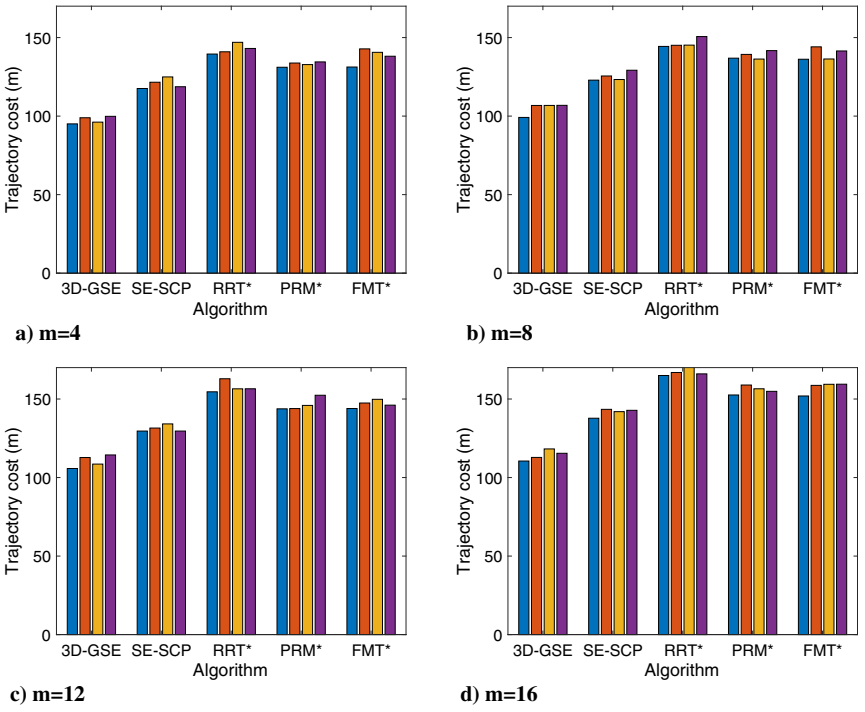


Fig. 12 Average trajectory cost with various algorithms.

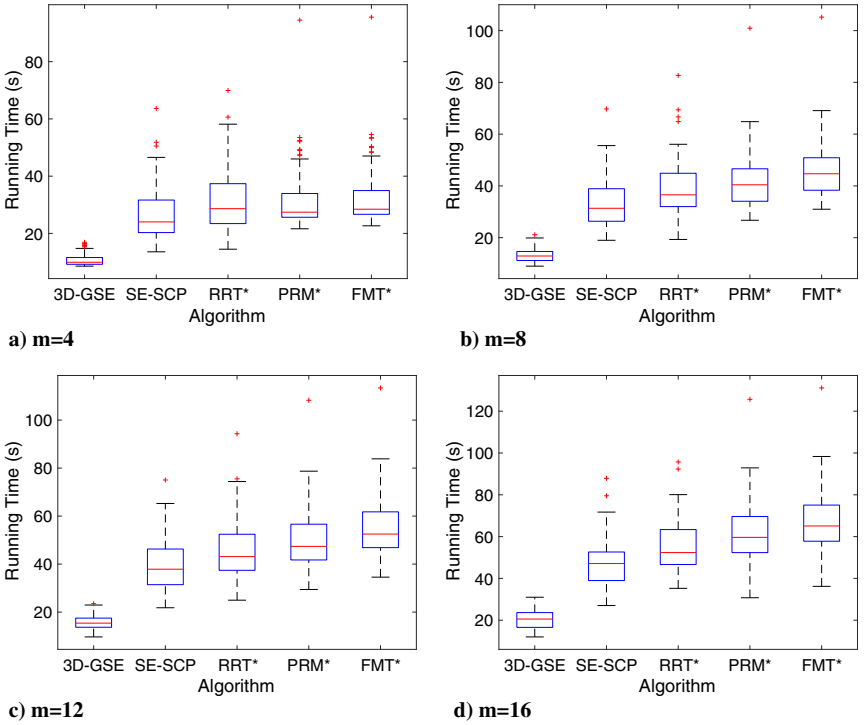


Fig. 13 Spread of running time with different numbers of obstacles.

draw random sample points from larger shapes even when there are nearby obstacles as compared to the SE-SCP algorithm, which creates smaller spheres when there are nearby obstacles because of its conservative spherical expansion feature.

As in Ref. [43], in this Note also, the SCP step has been used to generate the locally optimal trajectory within the set of all discovered homotopy classes (as discussed in Sec. II) that also satisfies actual UAV dynamics. For the simulation study in this Note, a multirotor UAV dynamics as given in Ref. [45] has been followed. Optimization over the last updated homotopy classes has been useful in reducing the computational burden to find a suboptimal trajectory. However, in the case of the 3D-GSE, since the discovered homotopy classes are obtained over paths containing vertices sampled from a larger shape (3-D generalized shape), the minimum trajectory cost is also found marginally lower than that in the cases of other algorithms.

V. Conclusions

In this Note, a novel motion-planning algorithm, termed 3D-GSE, has been presented for an unmanned aerial vehicle in a three-dimensional environment. The presented algorithm has mainly two steps. During the 3-D generalized shape expansion step, the algorithm

explores the workspace using random sampling from a 3-D generalized shape that represents the free space to a great extent. At the end of this step, geometric paths are obtained from the start position to the goal position. During the optimal trajectory generation step, the shortest path(s) is (are) generated first, and then a locally optimal trajectory is generated along the shortest path(s) using sequential convex programming over the updated homotopy class. Simulation studies in randomly generated obstacle-cluttered environments with low to high obstacle densities were presented to show that the optimal trajectory costs obtained by the 3D-GSE algorithm are marginally better than that of several existing seminal algorithms, whereas the computational burden of the 3D-GSE algorithm is significantly less than that of the other algorithms in comparison. This endows the practical implementablility of the presented 3D-GSE algorithm for UAV applications with limited computational resources for onboard computation, power, and sensing.

References

[1] Beard, R., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., and Goodrich, M., "Autonomous Vehicle Technologies for Small Fixed Wing UAVs," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 1, 2005, pp. 92–108. https://doi.org/10.2514/1.8371

[2] Winnefeld, J., and Kendall, F., "*Unmanned Systems Integrated Roadmap: FY2011-2036*," U.S. Dept. of Defense TR –11-S-3613, 2011.

[3] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, Vol. 2, IEEE Publ., Piscataway, NJ, 1985, pp. 500–505. https://doi.org/10.1109/ROBOT.1985.1087247

[4] Koren, Y., and Borenstein, J., "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *IEEE Conference on Robotics and Automation*, IEEE Publ., Piscataway, NJ, 1991, pp. 1398–1404. https://doi.org/10.1109/ROBOT.1991.131810

[5] Chakravarthy, A., and Ghose, D., "Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach," *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, Vol. 28, No. 5, 1998, pp. 562–574. https://doi.org/10.1109/3468.709600

[6] Chakravarthy, A., and Ghose, D., "Collision Cones for Quadric Surfaces," *IEEE Transactions on Robotics*, Vol. 27, No. 6, 2011, pp. 1159–1166. https://doi.org/10.1109/TRO.2011.2159413

[7] Chakravarthy, A., and Ghose, D., "Generalization of the Collision Cone Approach for Motion Safety in 3-D Environments," *Autonomous Robots*, Vol. 32, No. 3, 2012, pp. 243–266. https://doi.org/10.1007/s10514-011-9270-z

[8] Sunkara, V., Chakravarthy, A., and Ghose, D., "Collision Avoidance of Arbitrarily Shaped Deforming Objects Using Collision Cones," *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, 2019, pp. 2156–2163. https://doi.org/10.1109/LRA.2019.2900535

[9] Manathara, J., and Ghose, D., "Reactive Collision Avoidance of Multiple Realistic UAVs," *Aircraft Engineering and Aerospace Technology*, Vol. 83, No. 6, 2011, pp. 388–396. https://doi.org/10.1108/00022661111173261

[10] Tony, L., Ghose, D., and Chakravarthy, A., "Precision UAV Collision Avoidance Using Computationally Efficient Avoidance Maps," *AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum*, AIAA Paper 2018-0875, 2018. https://doi.org/10.2514/6.2018-0875

[11] Wilhelm, J., Clem, G., and Casbeer, D., "Circumnavigation and Obstacle Avoidance Guidance for UAVs Using Gradient Vector Fields," *AIAA Scitech 2019 Forum*, AIAA Paper 2019-1791, 2019. https://doi.org/10.2514/6.2019-1791

[12] Wilhelm, J. P., and Clem, G., "Vector Field UAV Guidance for Path Following and Obstacle Avoidance with Minimal Deviation," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 8, 2019, pp. 1848–1856. https://doi.org/10.2514/1.G004053

[13] Gong, Q., Lewis, L. R., and Ross, I. M., "Pseudospectral Motion Planning for Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 3, 2009, pp. 1039–1045. https://doi.org/10.2514/1.39697

[14] Upadhyay, S., and Ratnoo, A., "Smooth Path Planning for Unmanned Aerial Vehicles with Airspace Restrictions," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1596–1612. https://doi.org/10.2514/1.G002400

[15] Ghosh, S., Davis, D., and Chung, T., "A Guidance Law for Avoiding Specific Approach Angles Against Maneuvering Targets," *IEEE 55th Conference on Decision and Control*, IEEE Publ., Piscataway, NJ, Dec. 2018, pp. 4142–4147. https://doi.org/10.1109/CDC.2016.7798897

[16] Ghosh, S., Yakimenko, O. A., Davis, D. T., and Chung, T. H., "Unmanned Aerial Vehicle Guidance for an All-Aspect Approach to a Stationary Point," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 11, 2017, pp. 2871–2888. https://doi.org/10.2514/1.G002614

[17] Singh, S., and Sujit, P. B., "Landmarks Based Path Planning for UAVs in GPS-Denied Areas," *IFAC-PapersOnLine*, Vol. 49, No. 1, 2016, pp. 396–400. https://doi.org/10.1016/j.ifacol.2016.03.086

[18] Kabamba, P. T., Meerkov, S. M., and Zeitz, F. H., "Optimal Path Planning for Unmanned Combat Aerial Vehicles to Defeat Radar Tracking," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 2, 2006, pp. 279–288. https://doi.org/10.2514/1.14303

[19] Lu, P., and Liu, X., "Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 375–389. https://doi.org/10.2514/1.58436

[20] Sujit, P. B., Saripalli, S., and Sousa, J. B., "Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-Wing Unmanned Aerial Vehicles," *IEEE Control Systems Magazine*, Vol. 34, No. 1, 2014, pp. 42–59. https://doi.org/10.1109/MCS.2013.2287568

[21] Canny, J., "A Voronoi Method for the Piano-Movers Problem," *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, Vol. 2, IEEE Publ., Piscataway, NJ, 1985, pp. 530–535. https://doi.org/10.1109/ROBOT.1985.1087297

[22] Takahashi, O., and Schilling, R. J., "Motion Planning in a Plane Using Generalized Voronoi Diagrams," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 2, 1989, pp. 143–150. https://doi.org/10.1109/70.88035

[23] Dai, R., and Cochran, J., "Path Planning and State Estimation for Unmanned Aerial Vehicles in Hostile Environments," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 595–601. https://doi.org/10.2514/1.46323

[24] Asano, T., Asano, T., Guibas, L., Hershberger, J., and Imai, H., "Visibility-Polygon Search and Euclidean Shortest Paths," *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, IEEE Publ., Piscataway, NJ, 1985, pp. 155–164. https://doi.org/10.1109/SFCS.1985.65

[25] Alexopoulos, C., and Griffin, P. M., "Path Planning for a Mobile Robot," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 2, 1992, pp. 318–322. https://doi.org/10.1109/21.148404

[26] Maekawa, T., Noda, T., Tamura, S., Ozaki, T., and Machida, K.-I., "Curvature Continuous Path Generation for Autonomous Vehicle Using B-Spline Curves," *Computer-Aided Design*, Vol. 42, No. 4, 2010, pp. 350–359. https://doi.org/10.1016/j.cad.2009.12.007

[27] Maini, P., and Sujit, P. B., "Path Planning for a UAV with Kinematic Constraints in the Presence of Polygonal Obstacles," *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE Publ., Piscataway, NJ, 2016, pp. 62–67. https://doi.org/10.1109/ICUAS.2016.7502625

[28] Brooks, R. A., and Lozano-Perez, T., "A Subdivision Algorithm in Configuration Space for Findpath with Rotation," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 2, 1985, pp. 224–233. https://doi.org/10.1109/TSMC.1985.6313352

[29] Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, Vol. 1, No. 1, 1959, pp. 269–271. https://doi.org/10.1007/BF01386390

[30] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107. https://doi.org/10.1109/TSSC.1968.300136

[31] Howard, T. M., Green, C. J., and Kelly, A., "State Space Sampling of Feasible Motions for High Performance Mobile Robot Navigation in Highly Constrained Environments," *Field and Service Robotics*, Springer, Berlin, 2008, pp. 585–593. https://doi.org/10.1007/978-3-540-75404-6_56

[32] Yang, H., and Zhao, Y., "Trajectory Planning for Autonomous Aerospace Vehicles Amid Known Obstacles and Conflicts," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 6, 2004, pp. 997–1008. https://doi.org/10.2514/1.12514

[33] Mattei, M., and Blasi, L., "Smooth Flight Trajectory Planning in the Presence of No-Fly Zones and Obstacles," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 454–462. https://doi.org/10.2514/1.45161

[34] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, 1996, pp. 566–580. https://doi.org/10.1109/70.508439

[35] Karaman, S., and Frazzoli, E., "Sampling-Based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, Vol. 30, No. 7, 2011, pp. 846–894. https://doi.org/10.1177/0278364911406761

[36] LaValle, S. M., and Kuffner, J. J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, Vol. 20, No. 5, 2001, pp. 378–400. https://doi.org/10.1177/02783640122067453

[37] Janson, L., Schmerling, E., Clark, A., and Pavone, M., "Fast Marching Tree: A Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions," *International Journal of Robotics Research*, Vol. 34, No. 7, 2015, pp. 883–921. https://doi.org/10.1177/0278364915577958

[38] Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129. https://doi.org/10.2514/2.4856

[39] Bhatia, A., and Frazzoli, E., "Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems," *International Workshop on Hybrid Systems: Computation and Control*, Springer, Berlin, 2004, pp. 142–156. https://doi.org/10.1007/978-3-540-24743-2_10

[40] Cortés, J., Jaillet, L., and Siméon, T., "Molecular Disassembly with RRT-Like Algorithms," *IEEE International Conference on Robotics and Automation*, IEEE Publ., Piscataway, NJ, 2007, pp. 3301–3306. https://doi.org/10.1109/ROBOT.2007.363982

[41] Zucker, M., Kuffner, J., and Branicky, M., "Multipartite RRTs for Rapid Replanning in Dynamic Environments," *IEEE Conference on Robotics and Automation*, IEEE Publ., Piscataway, NJ, 2007, pp. 1063–1069. https://doi.org/10.1109/ROBOT.2007.363553

[42] Sun, C., Liu, Y.-C., Dai, R., and Grymin, D., "Two Approaches for Path Planning of Unmanned Aerial Vehicles with Avoidance Zones," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 8, 2017, pp. 2076–2083. https://doi.org/10.2514/1.G002314

[43] Baldini, F., Bandyopadhyay, S., Foust, R., Chung, S. J., Rahmani, A., Croix, J. P., Bacula, A., Chilan, C. M., and Hadaegh, F. Y., "Fast Motion Planning for Agile Space Systems with Multiple Obstacles," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2016-5683, 2016, pp. 1–14. https://doi.org/10.2514/6.2016-5683

[44] Zinage, V., and Ghosh, S., "An Efficient Motion Planning Algorithm for UAVs in Obstacle-Cluttered Environment," *Proceedings of American Control Conference*, IEEE Publ., Piscataway, NJ, July 2019, pp. 2271–2276.

[45] Wang, P., Man, Z., Cao, Z., Zheng, J., and Zhao, Y., "Dynamics Modelling and Linear Control of Quadcopter," *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, IEEE Publ., Piscataway, NJ, 2016, pp. 498–503. https://doi.org/10.1109/ICAMechS.2016.7813499