Polygon-Based Random Tree Search Algorithm for a Size-Changing Robot

Jangho Bae, Sumin Park, Mark Yim, and TaeWon Seo, *Senior Member, IEEE*

***Abstract*—This letter proposes the Polygon-based Random Tree search algorithm with Variable polygon size (PRT-V) for Variable Topology Truss (VTT). This algorithm improves on our previous path planning algorithm by using VTT's ability to change its size drastically. The algorithm partitions the space into unconstrained areas where normal (higher speed) locomotion can work using our previous technique and tighter space areas where the robot will need to shrink in size. We call the points that delimit these spaces *stopover points*. A Voronoi diagram-based roadmap method is used to automatically find appropriate stopover points. Based on the stopover points, the desired support polygons and the corresponding nominal lengths are generated by PRT-V. Then, the locomotion of VTT is generated by a non-impact locomotion algorithm to prevent tipping over. Simulations verify the performance of PRT-V in three different workspaces with obstacles. With PRT-V, VTT can squeeze through narrow passages or move efficiently in large spaces by increasing size, a unique feature.**

***Index Terms*—Motion and path planning, cellular and modular robots.**

I. Introduction

VARIABLE Geometry Truss (VGT) systems have a long history of research due to their flexibility [1], [2]. The main design feature of VGT systems is linearly actuated truss members and passive rotational joints that connect the truss members. Miura *et al.* proposed the concept of a VGT system and presented applications focused on fixed systems [3]. Research using VGT as a modular and mobile system has also been explored by Hamlin *et al.* [4]. Usevitch *et al.* designed and fabricated an octahedral-shaped isoperimetric soft robot [5].

We are developing the Variable Topology Truss (VTT) system to perform search and rescue operations in disaster sites. VTT is an improved VGT that can change the topology of the truss to be optimized for various tasks [6]. The hardware prototype of VTT is under development, including the topology reconfiguration mechanism [7], [8]. A spiral zipper actuator design with a very large extension ratio [9] is used for the linear actuators in VTT. This feature is beneficial for topology reconfiguration planning. This also gives VTT the ability to change its size on a large scale. We selected an octahedron as a topology for this research as shown in Fig. 1(a). The current hardware prototype is presented in Fig. 1(b).

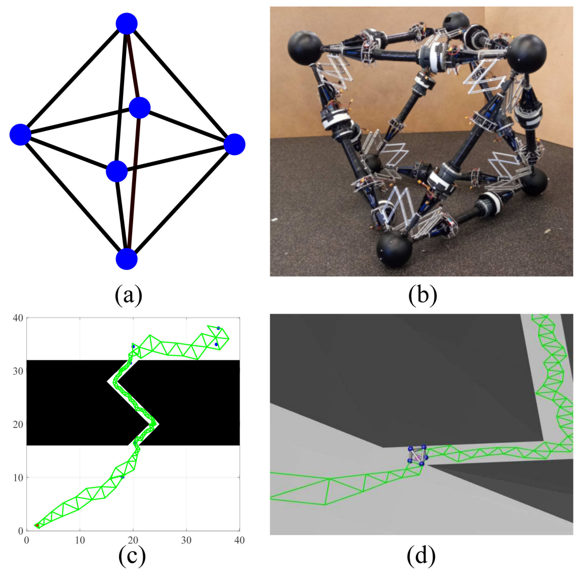

Fig. 1. The summary diagram of the letter: (a) Simplified diagram of octahedral structure, with nodes drawn as blue circles and members drawn as black lines; (b) Hardware prototype of VTT; (c) Support polygon trajectory generation with PRT-V; (d) Locomotion simulation.

Locomotion is the most important task for search and rescue operations. Our previous study proposed a Polygon-based Random Tree search (PRT) path planning for VTT [10]. The main idea of PRT is to pre-define support polygons with the Rapid Random Tree search algorithm [11]. The original PRT algorithm uses a fixed size support polygon. However, a large robot is ideal for fast and efficient movement in open areas, while a small robot is necessary to navigate narrow or cluttered areas. Successful search and rescue operations require good performance in both situations.

This letter presents the *Polygon-based Random Tree search algorithm with Variable polygon size* (PRT-V) for VTT systems. Here, the size of VTT can adapt to the environmental conditions. A roadmap method is included in the algorithm to find

Manuscript received February 23, 2021; accepted June 24, 2021. Date of publication July 7, 2021; date of current version July 6, 2022. This work was supported by the Industrial Core Technology Development Project through Ministry of Trade, Industry and Energy, South Korea (MOTIE) under Grant 11006-9072, the Fostering Global Talents for Innovative Growth Program (P0008748), and the Human Resource Development Program for Industrial Innovation(Global) (P0017306) supervised by the Korea Institute for Advancement of Technology. (KIAT). *(Co-first author: Jangho Bae and Simin Park)*, *(Corresponding author: TaeWon Seo.)*

Jangho Bae and Mark Yim are with the School of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19146 USA (e-mail: jhbae@rodel.snu.ac.kr; yim@grasp.upenn.edu).

Sumin Park is with the Samsung Electronics Co., Seoul 04763, Republic of Korea (e-mail: smpark@rodel.snu.ac.kr).

TaeWon Seo is with the School of Mechanical Engineering, Hanyang University, Seoul 04763, Republic of Korea (e-mail: taewon.seo1@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3095026, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3095026

characteristic points from the initial point to the goal. These key locations are called *stopover points*, which are used as reference points of path planning and size changing. We apply Voronoi diagram-based roadmap methods to find stopover points [12], [13]. Using PRT-V, VTT can squeeze into narrow passages or increase its size to move in open-space more efficiently as shown in Fig. 1(c) and (d).

This letter is organized as follows. In Section II, we analyze and present the design and kinematics of the VTT system. Subsequently, we present the PRT algorithm and nominal length changing method in Section III. The simulation results of the new algorithm are discussed in Section IV. Finally, the conclusion of the study is provided in Section V.

II. Variable Topology Truss System

A. Variable Topology Truss Overview

The VTT system can change the connectivity of the truss members depending on the assigned task. In this letter, we focus on the locomotion of VTT. A simple topology that has symmetry and can achieve smooth non-impact rolling locomotion is an octahedron. While a tetrahedron is theoretically simpler, the tight angle constraints at the nodes make it difficult to achieve smooth non-impact motion. The octahedron we use is shown in Fig. 1(a). The VTT topology has six vertices and twelve edges. We use the truss terminology and so refer to the edges of the graph of VTT as "members" and vertices of the graph of VTT as "nodes." Nodes link members with passive spherical joints. Each member contains a linear actuator for extending and retracting its length. To achieve a large extension ratio, we use a spiral zipper actuator for the members [9]. Theoretically, we can increase the extension ratio of a spiral zipper simply by using a longer zipper band. This feature allows VTT to change its size.

B. Kinematic Analysis

The notation used in this letter is defined as follows. The truss nodes are denoted as $\boldsymbol{N} = \{1, \ldots, 6\}$ and the truss members are denoted as $\boldsymbol{M} = \{m_1, \ldots, m_{12}\}$. The connectivity is expressed as $m_k = \{i, j\}$, which means k-th member connects node i and j. Two connected nodes are called "adjacent nodes." The vector $\boldsymbol{x}$ indicates position of all nodes as follows:

$$\boldsymbol{x} = [p_{1x}, \ldots, p_{6x}, p_{1y}, \ldots, p_{6y}, p_{1z}, \ldots, p_{6z}]^\top, \quad (1)$$

where position of i-th node is expressed as $\boldsymbol{P}_i = [p_{ix}, p_{iy}, p_{iz}]^\top$.

An analysis of the kinematics of VTT helps to understand the locomotion planning approach. The velocities between the projected center of mass position, $\boldsymbol{x}_C = [x_{Cx}, x_{Cy}]^\top$, and node positions is calculated as follows:

$$\dot{\boldsymbol{x}}_C = \mathbf{M}\dot{\boldsymbol{x}}, \quad (2)$$

where $\mathbf{M}$ is the center of mass Jacobian matrix.

The inverse kinematics of VTT are straightforward: we calculate the member length vector $\boldsymbol{L} = [L_1, \ldots, L_{12}]^\top$ using the node positions. The inverse kinematics can be calculated by differentiating the member length vector with respect to node position vector as follows:

$$\dot{\boldsymbol{L}} = \mathbf{R}\dot{\boldsymbol{x}}, \quad (3)$$

where $\mathbf{R}$ is is the Jacobian of the inverse kinematics.

Constraints are defined by considering the current hardware prototype design. The constraints represented in the simulation of this study are as follows: *Linear velocity limits*, *Angle limits between adjacent members*, *Manipulability lower limit*, and *Collision avoidance*. Detailed information of the constraints are provided in the previous letter [10].

III. Path Planning and Locomotion Algorithm

A. PRT-V Algorithm Overview

The proposed algorithm changes the size of VTT with respect to the environment as needed. The overview of PRT-V algorithm is summarized in Algorithm 1. The initial configuration of the VTT is given by a starting support polygon $Polygon_{init}$ and its center, $\boldsymbol{S}_{ini}$. The goal position is denoted by $\boldsymbol{S}_{fin}$. Obstacles are expressed as $Q = \{Q_1, \ldots, Q_m\}$.

First, the algorithm finds key locations between the initial point and the goal by analyzing the environment. The PRT-V algorithm characterizes the workspace by creating a Voronoi diagram. The shortest path through the diagram, $V = \{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_n\}$, is used as a rough trajectory for the rest of the algorithm. Several key locations, called stopover points, indicate the points that require size adjustment. These stopover points, denoted as $S = \{\boldsymbol{S}_0, \ldots, \boldsymbol{S}_n\}$, are selected along the Voronoi path, and a desired nominal length $\hat{L}_{des}$ is calculated for each one.

Finally, the stopover points are connected by support polygons that have the correct size. The support polygon trajectory is generated by the random tree search algorithm, $PolygonSearch$. In each step of the search, a nominal length $\hat{L}$ is used as the base length for generating support polygons. $PolygonSearch$ returns a set of trees T, each of which contains a set of support polygons.

Algorithm 1: Polygon-based Random Tree search with Variable polygon size (PRT-V).

```
Input: $Polygon_{init}$, $Q$, $\boldsymbol{S}_{ini}$, $\boldsymbol{S}_{fin}$
1 $V = VoronoiPath(Q)$;
2 $S = FindStopoverPoints(V, Q, \boldsymbol{S}_{ini}, \boldsymbol{S}_{fin})$;
3 $\hat{L}_{des} = DesiredNominalLengths(S, Q)$;
  $T = PolygonSearch(Polygon_{init}, S, \hat{L}_{des})$;
5 return $T$;
```

B. Workspace Analysis

The workspace analysis determines when size-changing is required and how much. The workspace and the obstacle data (including shape and position) are known a priori. Fig. 2(a) shows an example workspace with several obstacles. Boundaries of the workspace are denoted as red dotted lines, and obstacles are drawn as black-filled closed curves.

A generalized Voronoi diagram can be used for motion planning [12]–[14]. Here, we generate the Voronoi diagram from a set of reference points. The reference points comprise points around the obstacles and along a virtual border defined by

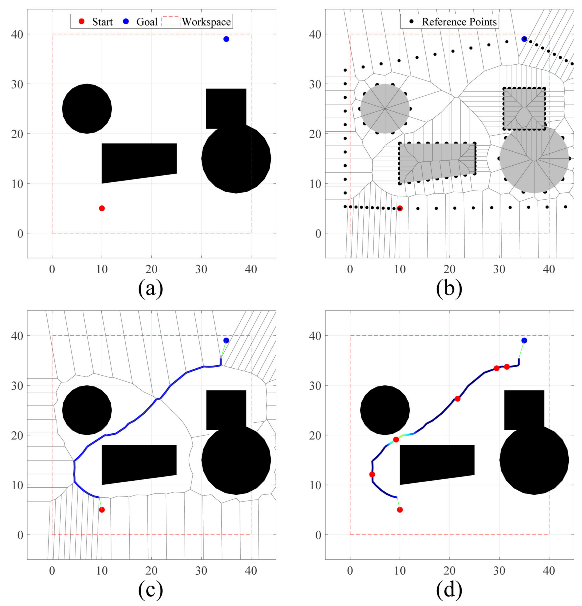

Fig. 2. Process of analyzing a workspace: (a) A workspace is shown with initial point, the goal, and obstacle data; (b) A Voronoi diagram is generated; (c) The rough trajectory to the goal is generated by finding the shortest path on Voronoi diagram. Voronoi line segments intersecting obstacles are removed. The Voronoi Path is shown in blue, the nearest point from the initial point and the goal in green; (d) Stopover points are selected based on the minimum distance from obstacles. The color of the Voronoi path indicates the distance to the nearest obstacle; darker is farther.

the convex hull of the initial point, goal point, and a padded rectangular bounding box that encloses all obstacles. Fig. 2(b) shows the raw Voronoi diagram from an example workspace. Then, the Voronoi line segments that intersect obstacles are deleted. Using Dijkstra's algorithm, the shortest path from the start point to the endpoint is chosen as a rough trajectory. This process is expressed as $VoronoiPath$ in Algorithm 1. Fig. 2(c) shows the rough trajectory derived from the Voronoi diagram.

Algorithm 2: FindStopoverPoints.

```
   Input: $V = \{\boldsymbol{V}_1, \cdots, \boldsymbol{V}_n\}$, $Q$, $\boldsymbol{S}_{ini}$, $\boldsymbol{S}_{fin}$
1  $\boldsymbol{S}_0 = \boldsymbol{S}_{ini}$;
2  $\boldsymbol{V}_{Temp} = \boldsymbol{V}_1$;
3  for $i = 2$ to $length(V)$ do
4      $\Delta d_{min,i} = d_{min}(\boldsymbol{V}_i, Q) - d_{min}(\boldsymbol{V}_{i-1}, Q)$;
5      if $sign(\Delta d_{min,i}) \neq sign(\Delta d_{min,i-1})$ then
6          if $\delta(\boldsymbol{V}_i) - \delta(\boldsymbol{V}_{Temp}) \geq Threshold$ then
7              $S.AddPoint(\boldsymbol{V}_i, \boldsymbol{V}_{Temp})$;
8          end
9          $\boldsymbol{V}_{Temp} = \boldsymbol{V}_i$;
10     end
11 end
12 $S.RemoveClosePoint()$;
13 $S.Add(\boldsymbol{S}_{fin})$;
14 return $S$;
```

Stopover points are bottleneck points along the Voronoi path. The selection criteria finds local minima or maxima based on the distance to obstacles, with a threshold to prevent stopover points from clustering too close together.

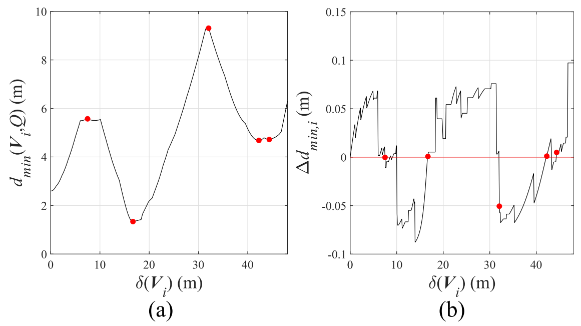

Fig. 3. Determining stopover points for path planning: (a) The minimum distance from obstacles d_{min} versus travel distance; (b) Δd_{min} versus travel distance. The stopover points are denoted as red dots in the figure.

Algorithm 2 explains this method denoted as $FindStopoverPoints$ in Algorithm 1. It includes the minimum distance from obstacles to points on the Voronoi path, $d_{min}(\boldsymbol{V}_i, Q)$ and the change of $d_{min}(\boldsymbol{V}_i, Q)$, $\Delta d_{min,i}$. The algorithm detects sign changes of $\Delta d_{min,i}$. If the distance from the previous sign change point $\boldsymbol{V}_{Temp}$ to the current sign change point $\boldsymbol{V}_i$ is larger than a threshold, both points are included as stopover points. The threshold value is set as a small value when $d_{min}(\boldsymbol{V}_i, Q)$ is small, and vice versa. The distance from the start point $\boldsymbol{V}_1$ to $\boldsymbol{V}_i$ along the Voronoi path is denoted as $\delta(\boldsymbol{V}_i)$.

After finding stopover points, the stopover points that are too close together are removed in $RemoveClosePoint$ on line 11, Algorithm 2. If two stopover points are too close, the nominal length may not be reached in time. In the shrinking case, this may result in the truss colliding with obstacles. $RemoveClosePoint$ removes the stopover point where the next stopover point has a smaller desired nominal length and the distance d_{thr} is not sufficient. The calculation of d_{thr} is shown at the end of this section.

Fig. 3 illustrates the stopover point algorithm. The distance data versus $\delta(\boldsymbol{V})$ is denoted in Fig. 3(a), and the change ratio is denoted in Fig. 3(b). These stopover points are also presented in Fig. 2(d).

C. Support Polygon Search Algorithm

A support polygon is the polygon defined by the ground contact points. VTT should follow the desired support polygons to prevent large distortion, as stated in previous work [10]. In this letter, the desired support polygons are derived by performing a modified Polygon-based Random Tree search (PRT) from one stopover point to the next. We improved PRT to change VTT's nominal length. This improvement allows VTT to adjust its size to be small enough to avoid obstacle collision yet large enough to move efficiently.

The Polygon-based Random Tree search algorithm with Variable polygon size (PRT-V) is summarized in Algorithm 3. PRT-V produces desired support polygons from the previous stopover point to the next one.

Algorithm 3: PolygonSearch.

```
Input: $Polygon_{init}$, $S$, $\hat{L}_{des}$
1  $T_1.init(Polygon_{init})$;
2  for $i = 1$ to $length(S)$ do
3      $Goal\_Position = \boldsymbol{S}_i$;
4      $Reach\_Goal = false$;
5      while $Reach\_Goal == false$ do
6          if $Random[0,1] \geq MixingFactor$ then
7              $Obj\_Point = Random\_Position$;
8          else
9              $Obj\_Point = Goal\_Position$;
10         end
11         $FootAlts = \{\}$;
12         for $Polygon$ in $T_i$ do
13             $\hat{L} = NextNominalLength(...$
               $Polygon, \boldsymbol{S}_i, \hat{L}_{des,i}, \hat{L}_{des,i-1})$;
14             $FootAlts.Add(MakeFoots(...$
               $Polygon, \hat{L}, Obj)$;
15         end
16         $\boldsymbol{Foot}_{new} = FootAlts.ClosestTo(Obj\_Point)$;
17         $Polygon_{new} = MakePolygon(\boldsymbol{Foot}_{new}, \hat{L})$;
18         if $CollisionFree(Polygon_{new})$ then
19             $T_i.AddPolygon(Polygon_{new})$;
20             if $Goal\_Position \in Polygon_{new}$ then
21                 $Reach\_Goal = true$;
22             end
23         end
24     end
25     $T_{i+1}.init(Polygon_{new})$;
26 end
27 return $T$;
```

For every step of PRT-V, an objective point is selected randomly within a workspace. A function $Random\_Position$ on line 7, Algorithm 3 returns a random point within a given workspace uniformly. There is a chance that the objective point is located at the next stopover point. This probability is expressed as a mixing factor and set as 30% in this study. The objective point is used as an indicator for determining the next support polygon.

The PRT-V algorithm builds a tree structure T of support polygons to explore the space and find a path to the goal. Each $Polygon$ in the tree consists of the three vertices' position vectors, $\boldsymbol{Foot}_i$, and its nominal length $\hat{L}$.

$$Polygon = \{\boldsymbol{Foot}_1,\ \boldsymbol{Foot}_2,\ \boldsymbol{Foot}_3,\ \hat{L}\}. \quad (4)$$

A new $Polygon$ is generated from its parent $Polygon$ by connecting two $Foot$s of the parent $Polygon$ and one new $Foot$. Fig. 4 summarizes the process of producing a $Polygon$ from a parent $Polygon$. Every time an objective point is selected, three $Foot$ alternatives (generated by $MakeFoots$) are placed from the three edges of each $Polygon$ in the tree, using the nominal length ($\hat{L}$) generated from that $Polygon$. The closest $Foot$ alternative to the objective point is selected to create the next $Polygon$ to be added to the tree. After that, a small distortion is applied to give flexibility for planning. Within a circular region around the $Foot$ alternative, the next $Foot$ is selected to have

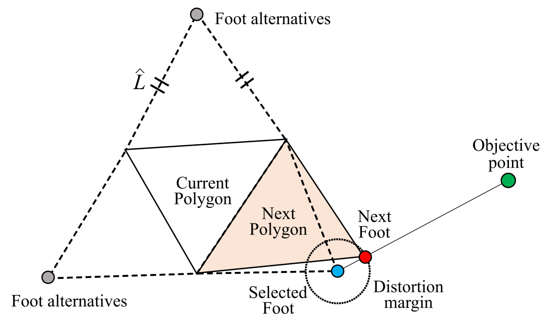

Fig. 4. Generation of the new support polygon from all previous ones. (Finding $\boldsymbol{Foot}_{new}$ in Algorithm 3) The new $Foot$ for producing next $Polygon$ is denoted as a red dot.

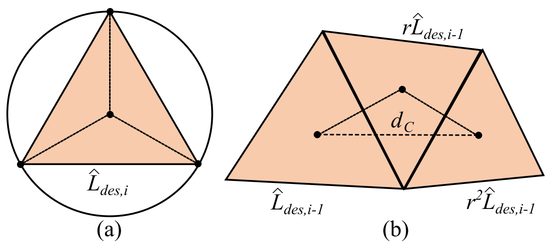

Fig. 5. Parameter criterion for determining $\hat{L}_{des,i}$ (a) A circumcircle of regular triangle with edge length $\hat{L}_{des,i}$; (b) Displacement of polygon center caused by steps during changing $\hat{L}$ from $\hat{L}_{des,i-1}$ with ratio r.

a minimum distance from the objective point. The radius of the circular region is called the distortion margin. Finally, a new polygon is constructed using this $Foot$ and added to the tree. PRT-V repeats this process until the next $Polygon$ covers a stopover point. If the number of loop iterations in Algorithm 3 is larger than the maximum limit, the algorithm resets the current T and starts from the stopover point again. (This is omitted in Algorithm 3 for simplicity).

The overall size of the VTT is determined from the size of the support polygon. The nominal length $\hat{L}$ is the base length for generating desired support polygons of VTT. Unlike our previous algorithm, $\hat{L}$ varies along the course of the path. Each stopover point $\boldsymbol{S}_i$ has an associated desired nominal length $\hat{L}_{des,i}$, which is related to how close that point is to any obstacles. We calculate this desired nominal length for each stopover point by finding the edge length of a triangle inscribed in a circle with its radius determined by the obstacle distance as shown in Fig. 5(a).

$$\hat{L}_{des,i} = \min(\alpha\sqrt{3}d_{min}(\boldsymbol{S}_i, Q),\ \hat{L}_{max}), \quad (5)$$

where $d_{min}(\boldsymbol{S}_i, Q)$ is the minimum distance between $\boldsymbol{S}_i$ and all obstacles Q. The parameter α is arbitrary constant to provide a buffer around the obstacles. In this study α is set to 0.5. An upper limit for $\hat{L}_{des,i}$, $\hat{L}_{max}$, is set as 2.5 m. Equation (5) is used for implementing function $DesiredNominalLengths$ in Algorithm 1.

PRT-V includes an algorithm to adjust the nominal member length. To prevent constraint violation from the physical system, the algorithm smoothly changes the nominal length between stopover points. As stated earlier, each stopover point $\boldsymbol{S}_i$ has an associated desired nominal length, $\hat{L}_{des,i}$. The truss also starts with an initial desired nominal length, $\hat{L}_{des,0}$, which is determined by the obstacle distance at $\boldsymbol{S}_{ini}$. As the truss moves

Algorithm 4: *NextNominalLength.*

```
   Input: $Polygon$, $\boldsymbol{S}_i$, $\hat{L}_{des,i}$, $\hat{L}_{des,i-1}$
 1 $r = CalculateRatio(\hat{L}_{des}, \hat{L}_{des,i-1})$;
 2 $d_{thr} = CalculateThresholdDistance(r, \hat{L}_{des,i-1})$;
 3 $\hat{L}_{pre} = Polygon.NomLength$;
 4 if $Distance(Center(Polygon), \boldsymbol{S}_i) \geq d_{thr}$ then
 5     $\hat{L} = Adjust(\hat{L}_{pre}, \hat{L}_{des,i-1})$;
 6 else
 7     if $\hat{L}_{pre}$ reaches or overshoots $\hat{L}_{des,i}$ then
 8         $\hat{L} = \hat{L}_{des,i}$;
 9     else
10         $\hat{L} = r\hat{L}_{pre}$;
11     end
12 end
13 return $\hat{L}$;
```

from $\boldsymbol{S}_{i-1}$ to $\boldsymbol{S}_i$, we need to decide the nominal length $\hat{L}$ at each polygon step such that $\hat{L}$ transitions from $\hat{L}_{des,i-1}$ to $\hat{L}_{des,i}$. This functionality is implemented by $NextNominalLength$, which is summarized in Algorithm 4.

We assume that at the start of the trajectory, the environment is still characterized by $\hat{L}_{des,i-1}$, so $\hat{L}$ continues to track $\hat{L}_{des,i-1}$. Once the truss gets closer to $\boldsymbol{S}_i$, it begins to transition to $\hat{L}_{des,i}$. The important parameters are the distance at which this process starts, d_{thr}, and the incremental size change ratio, r.

Because most constraints are related to angles, it is better to change the nominal length proportionally at each step. Given $\hat{L}_{des,i-1}$ and $\hat{L}_{des,i}$, we can transition from one to the other over N_{step} steps by multiplying $\hat{L}$ by the following ratio r:

$$r = \left(\frac{\hat{L}_{des,i}}{\hat{L}_{des,i-1}}\right)^{\frac{1}{N_{step}}}, \tag{6}$$

N_{step} is set as 6 initially. However, if r is too big because of a considerable length change, N_{step} is increased until r enters the range $(0.75, 1.25)$. This procedure is implemented in the $CalculateRatio$ function in Algorithm 4.

If we change $\hat{L}$ from $\hat{L}_{des,i-1}$ by the ratio r for N_{step} steps, the total displacement of the $Polygon$ center can be approximated as follows:

$$d_C = \frac{1}{4}(1+r)\frac{r^{N_{step}}-1}{r-1}\hat{L}_{des,i-1}. \tag{7}$$

This situation is presented in Fig. 5(b). By applying a margin on d_C, we define the threshold distance $d_{thr} = kd_C$. The threshold margin k is set as 1.5.

Once the current $Polygon$ center is closer to the next stopover point than d_{thr}, we begin to multiply $\hat{L}$ by r at every subsequent step. This incremental change is applied each step until the truss nominal length $\hat{L}$ reaches or overshoots $\hat{L}_{des,i}$. After this point, $\hat{L}$ tracks $\hat{L}_{des,i}$ until the truss arrives at $\boldsymbol{S}_i$. In some cases, the truss nominal length may not successfully reach $\hat{L}_{des,i}$ before it arrives at $\boldsymbol{S}_i$. In this case, the truss will attempt to reach the desired nominal length during the first portion of the next trajectory. This special case is handled by the $Adjust$ function.

In some cases, the support polygon planning may not be able to pass a narrow area entrance, even if the size is small enough

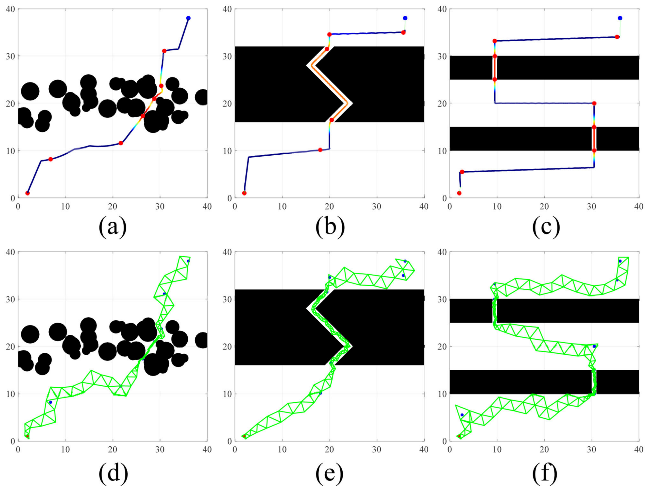

Fig. 6. Support polygon planning simulations in three different workspaces. The color of Voronoi path indicates the minimum distance from obstacles. For workspace 1: (a) Voronoi path and stopover points; (d) Desired support polygons. For workspace 2: (b) Voronoi path and stopover points; (e) Desired support polygons. For workspace 3: (c) Voronoi path and stopover points; (f) Desired support polygons.

because the orientation of the support polygon limits the motion of the next step. PRT-V goes back to the previous stopover point to prevent this situation if the while loop in Algorithm 3 repeatedly reaches the maximum iteration limit after a certain stopover point.

IV. Simulation Results

To verify the performance of the PRT-V algorithm, we select three workspaces with different kinds of obstacles. For all three examples, the limits are 0 m to 40 m for the x-direction and 0 m to 40 m for the y-direction. The initial position is (2 m, 1 m), and the goal is located at (36 m, 38 m). Workspace 1 contains forty randomly generated circular obstacles. By randomly selecting obstacles, difficult conditions may appear that the authors do not expect. The obstacle centers are distributed throughout positions (0 m, 15 m) to (40 m, 25 m), and the radii are distributed from 0.8 m to 2 m. Workspace 2 has one zigzag-shaped narrow passage between two large obstacles. This tests PRT-V against long narrow passages. The width of the passage is 2 m. Workspace 3 has two separate narrow passages. With this environment, we can test if PRT-V can handle repeated size changes. Both passages are 1m wide. They are 20 m away from each other in the x-direction and 10 m away in the y-direction. The initial edge length of VTT is set as 1.25 m. The three examples are presented in Fig. 6. Black-filled shapes denote obstacles. The initial point is a red dot on the left bottom corner, and the goal is a blue dot.

To follow the desired support polygons, we used non-impact rolling locomotion [15] as used in previous work with PRT [10]. In the ‘moving phase,’ The VTT center of mass moves through the line that connects Polygon centers. When the VTT center of mass reaches near a support polygon boundary, the control is changed to a ‘landing phase.’ After the front node touches the ground, VTT switches back to the moving phase again.

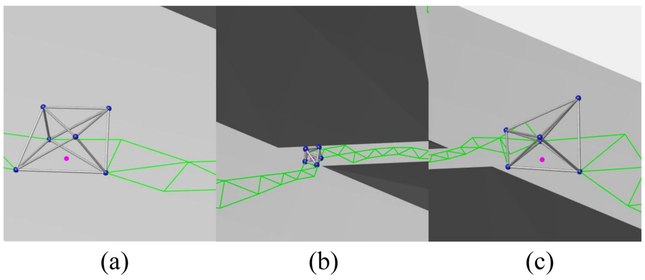

Fig. 7. Locomotion simulation of VTT in workspace 2: (a) Moving in a large space; (b) Retracting size to go into a narrow passage; (c) Increasing size again after the passage.

TABLE I
COMPARISON WITH THE PREVIOUS ALGORITHMS

	PRT	PRT w/ stopover	PRT-V
Success ratio	89%	92%	98%
Running time (sec)	268.3(311.5)	237.2(204.6)	128.2(191.2)
Total steps	548.9(34.4)	576.3(28.6)	91.0(23.6)

A detailed locomotion algorithm is presented in our previous studies [10], [15].

Fig. 6 shows PRT-V simulation results of roadmaps and support polygons. The Voronoi diagram based roadmap method successfully generates proper stopover points for all three workspaces. Using stopover points helps PRT-V stably generate the support polygon path. By changing the size of its support polygon, VTT finds trajectories that would not be valid if constrained to its original size. The randomness of PRT-V is beneficial for rapidly finding passages through complex workspaces 1 and 2. PRT-V successfully created motion plans for all cases.

Fig. 7 shows the locomotion simulation result on workspace 2. The support polygon generation and locomotion simulations are presented in the supplementary video.

In Table I we compare PRT-V algorithm with the two other versions of previous PRT algorithms to prove the advantages of the Voronoi-based roadmap method and size-changing feature. The first one is the PRT algorithm, which maintains the fixed nominal length and uses only the goal position for path planning. The second one is PRT algorithm adding stopover points generated by Voronoi roadmap method. The third one in the table is the proposed PRT-V algorithm. In this comparison, the fixed nominal length of PRT and PRT with stopover points are set with a sufficiently small value, 0.2 m. Workspace 1 is used for the reference environment to test 20 different random conditions. For each, we show the calculation time, total steps to reach the goal, and the success ratio. We define planning failure as a running time that exceeds 20 minutes. Results are listed as the average of 100 simulations with the standard deviation in parentheses.

Applying the roadmap method and size-changing feature significantly improves the performance. Most critical is decreasing the number of steps as it lowers the number of support polygon transitions. This makes locomotion more safe and efficient. Calculation time improves as the iterations are reduced. Decreasing calculation time also increases the success ratio due to the way success is defined. The guiding direction from the roadmap method slightly decreases the calculation time for PRT as well. However, without allowing size-changing, the roadmap method may select impossible stopover points in extreme cases.

V. CONCLUSION

In this letter, we show a path planning algorithm, PRT-V, that includes size changing of the VTT system. Because of the large extension ratio of VTT actuators, it is possible to drastically change the size of VTT. Using PRT-V, the support polygon with proper size can be generated automatically with respect to environmental conditions. The simulation results show that VTT can go through very narrow passages, yet also maintain more efficiently sized gaits in open areas which is impossible for conventional robots. While the random search algorithm used for finding support polygon trajectory succeeded, no optimization was attempted in this work. This is left for the future as is the optimization of the parameters of PRT-V for other types of workspaces.

REFERENCES

[1] S. Curtis *et al.*, “Tetrahedral robotics for space exploration,” in *Proc. IEEE Aerosp. Confe.*, 2007, pp. 1–9.
[2] P. C. Hughes, W. G. Sincarsin, and K. A. Carroll, “Trussarm-a variable-geometry-truss manipulator,” *J. Intell. Mater. Syst. Structures*, vol. 2, no. 2, pp. 148–160, 1991.
[3] K. Miura, H. Furuya, and K. Suzuki, “Variable geometry truss and its application to deployable truss and space crane arm,” *Acta Astronautica*, vol. 12, no. 7-8, pp. 599–607, 1985.
[4] G. J. Hamlin and A. C. Sanderson, “Tetrobot: A modular approach to parallel robotics,” *IEEE Robot. Automat. Mag.*, vol. 4, no. 1, pp. 42–50, 1997.
[5] N. S. Usevitch, Z. M. Hammond, M. Schwager, A. M. Okamura, E. W. Hawkes, and S. Follmer, “An untethered isoperimetric soft robot,” *Sci. Robot.*, vol. 5, no. 20, eaaz0492, 2020.
[6] A. Spinos, D. Carroll, T. Kientz, and M. Yim, “Variable topology truss: Design and analysis,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2717–2722.
[7] E. Park, J. Bae, S. Park, J. Kim, M. Yim, and T. Seo, “Reconfiguration solution of a variable topology truss: Design and experiment,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1939–1945, Apr. 2020.
[8] S. Park, “Stable rolling locomotion for variable topology truss robot,” Ph.D. dissertation, Seoul National University, 2020.
[9] F. Collins and M. Yim, “Design of a spherical robot arm with the spiral zipper prismatic joint,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 2137–2143.
[10] S. Park, J. Bae, S. Lee, M. Yim, J. Kim, and T. Seo, “Polygon-based random tree search planning for variable geometry truss robot,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 813–819, Apr. 2020.
[11] S. M. LaValle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” Technical Report, Computer Science Department, Iowa State University (TR 98-11), 1998.
[12] O. Takahashi and R. J. Schilling, “Motion planning in a plane using generalized voronoi diagrams,” *IEEE Trans. Robot. Automat.*, vol. 5, no. 2, pp. 143–150, Apr. 1989.
[13] M. Šeda and V. Pich, “Robot motion planning using generalised voronoi diagrams,” in *Proc. 8th WSEAS Int. Conf. Signal Process., Comput. Geometry Artif. Vis.*, 2008, pp. 215–220.
[14] D.-T. Lee and R. L. Drysdale, III, “Generalization of voronoi diagrams in the plane,” *Soc. Ind. Appl. Math. J. Comput.*, vol. 10, no. 1, pp. 73–87, 1981.
[15] S. Park, E. Park, M. Yim, J. Kim, and T. Seo, “Optimization-based nonimpact rolling locomotion of a variable geometry truss,” *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 747–752, Apr. 2019.