IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Polygon-based Random Tree Search Planning for Variable Geometry Truss Robot*

Sumin Park[1], Jangho Bae[2], Seohyeon Lee[3], Jongwon Kim[1], Mark Yim[2] and TaeWon Seo[3], *Member, IEEE*

***Abstract*— This paper proposes the use of a polygon-based random tree path planning algorithm for a variable geometry topology system (VGT). By combining a path planning algorithm and our previous non-impact locomotion algorithm, the proposed VGT system reaches an objective point. The proposed path planning algorithm provides the desired set of support polygons with a modified rapid random tree algorithm. The algorithm can significantly reduce distortion of the VGT system while moving by limiting the deformation of the desired support polygon. With this algorithm feature, constraint violations of the system were significantly reduced with respect to using a normal rapid random tree algorithm for path planning. The performance of the algorithm was validated using the simulation results.**

I. INTRODUCTION

A variable geometry topology (VGT) is a truss structured system that comprises actively actuated linear trusses linked with passive rotational ball joints. The VGT concept has been researched for many years. VGT was first proposed for the design of a space crane arm by Miura et al. [1]. Hughes et al. also designed a space manipulator, Trussarm, using the VGT concept [2]. The VGT concept has not only been used for a fixed manipulator arm but also for a locomotion system. Given the flexibility of VGT, it can adapt to complexly shaped terrain. Hamlin and Sanderson developed a reconfigurable VGT for investigating various surfaces with a six-legged walking configuration [3]. Curtis et al. developed and fabricated various VGT designs for space exploration [4]. We are currently developing a VGT system for performing search and rescue tasks at disaster sites [5]. The flexible properties of the VGT have advantages when moving on irregular terrain at disaster sites.

To facilitate VGT system moving on terrain, the locomotion algorithm and shape of the system should be derived. Previous VGTs have used various locomotion algorithms for moving. Hamlin and Sanderson introduced a six-legged walking gait motion for VGT [3]. Lee and Sanderson proposed a locomotion algorithm that used a tipping and rolling motion for icosahedral VGT [6]. Usevitch et al. improved the tipping and rolling locomotion algorithm by adding a shape-changing task and making the center of mass follow a desired trajectory [7]. These previous dynamic rolling locomotion approaches including tipping motion may cause structural damage to the system because of the impact force from collision with the surface. We introduced a non-impact rolling locomotion algorithm for VGT to make its center of mass follow the desired trajectory [8]. With this algorithm, the system can perform a rolling motion without tipping by maintaining the center of mass inside the support polygon. However, a large-scale path planning algorithm is needed to automatically perform search and rescue operation.

The rapidly-exploring random tree (RRT) search algorithm is among the most popular methods for large-scale path planning of a robot. RRT was first proposed by Lavalle to solve a high degree-of-freedom problem with nonholonomic constraints [9]. Path planning with an obstacle is a typical case of having a high degree-of-freedom and nonholonomic constraints. RRT implementation in path planning for mobile robots was introduced by Bruce and Veloso [10]. The RRT method has advantages for planning a path on an irregular plane and with uncertainties as shown in Bry and Roy [11]. We selected and modified the RRT algorithm for large-scale path planning at disaster site.

In this paper, the polygon-based random tree (PRT) path planning algorithm is proposed for VGT. Previous locomotion algorithms only considered the trajectory of the center of mass, which cannot guarantee the next rolling step of the VGT. A large amount of distortion from the original shape typically caused constraint violations. Therefore, we developed a path planning algorithm that can guarantee the next rolling step without violating hardware constraints by maintaining the shape of the VGT support polygon. The proposed PRT provides the desired position and shape set of the support polygons to reach the objective point. The VGT follows the set of support polygons using our non-impact rolling locomotion algorithm [8]. The advantages of PRT were demonstrated by simulation results.

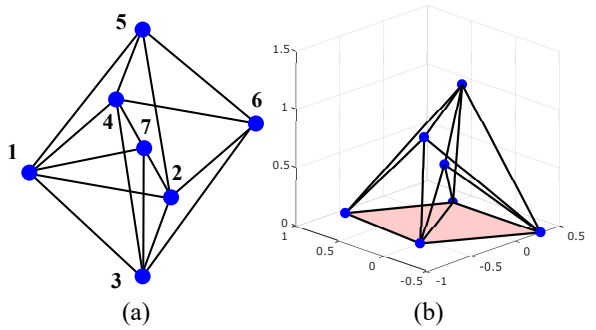
This paper is organized as follows. In section 2, kinematic analysis of our VGT is introduced. Support polygon-based path planning and non-impact rolling locomotion algorithm are presented in section 3. Algorithm simulation results are provided in section 4. Finally, conclusions are summarized in section 5.

*This work was supported by the Industrial Core Technology Development Project through Ministry of Trade, Industry and Energy, South Korea (MOTIE) under Grant 1006-9072 and the Fostering Global Talents for Innovative Growth Program (P0008748, Global Human Resource Development for Innovative Design in Robot and Engineering) supervised by the Korea Institute for Advancement of Technology. (KIAT) *(Co-corresponding author: Mark Yim and TaeWon Seo)*, *(Co-first authors: Sumin Park, Jangho Bae)*

[1]S. Park, and J. Kim are with the Department of Mechanical Engineering, Seoul National University, Seoul 08826, Republic of Korea smpark@rodel.snu.ac.kr; jongkim@snu.ac.kr

[2]J. Bae and M. Yim are with the School of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19146 USA jangho.bae91@gmail.com; yim@seas.upenn.edu

[3]S. Lee and T. Seo are with the School of Mechanical Engineering, Hanyang University, Seoul 04763, Republic of Korea hotkowork@gmail.com; taewonseo@hanyang.ac.kr *S. Lee is currently working as a visiting researcher at University of Pennsylvania.*



(a) (b)

Fig. 1. Structure of the proposed octahedral VGT: (a) initial configuration and node numbers and (b) configuration during rolling motion. The VGT support polygon is denoted in pink.

II. SYSTEM ANALYSIS

A. Configuration and Notation

Our proposed VGT system was designed to perform search and rescue at disaster sites. We considered topology reconfiguration to achieve both movement and structure shoring [5]. By minimizing the number of edges and vertexes, a modified octahedral shape was created. Fig. 1 shows the VGT configuration diagram. The VGT has 7 vertexes and 16 edges; one vertex is in the center of the VGT and the others are around the octahedron. The center vertex is connected to the four vertexes around the octahedron to maintain its position.

In the remainder of the paper, the following terms are used. Each VGT vertex was termed a *node*, and each edge was termed a *member*. A node comprises a rotational 3-degrees-of-freedom passive joint, while a member is operated using a linear actuator. A polygon that comprises nodes attached to the ground is termed a *support polygon*, and is denoted in pink in Fig. 1. The connection between nodes and members were defined by a graph $\boldsymbol{G}$ that consists of a node set $\boldsymbol{N} = \{1, 2, \cdots, 7\}$ and a member set $\boldsymbol{M} = \{m_1, m_2, \cdots, m_{16}\}$, where $m_k = \{i, j\}$ is a member that connects the i-th and j-th nodes. The position vector of the i-th node is denoted as $\boldsymbol{P}_i = [p_{ix}, p_{iy}, p_{iz}]^\top$. The vector $\boldsymbol{x}$ contains all position vectors of the nodes $\boldsymbol{x} = [p_{1x}, \cdots, p_{7x}, p_{1y}, \cdots, p_{7y}, p_{1z}, \cdots, p_{7z}]^\top$. Two nodes that are connected to each other by a member are termed *adjacent nodes*.

B. Kinematics Analysis

VGT kinematic equations were derived to define VGT movement. First, the relation between the projected center of mass position and node position was derived. The mass values of nodes are considered to be all equal and the mass of members is neglected. By differentiating the relation equation, the Jacobian for the projected center of mass position can be derived as follows:

$$\dot{\boldsymbol{x}}_{CM} = \mathbf{M}\dot{\boldsymbol{x}} \tag{1}$$

where $\boldsymbol{x}_{CM} = [x_{CM}, y_{CM}]^\top$ is the center of mass position projected to the ground without z component, and $\mathbf{M}$ is the Jacobian matrix of center of mass.

Second, the relation between the length of members and position of nodes was derived, i.e. the inverse kinematics of the VGT. The length of the k-th member that connects node i and j can be written as (2). By combining the length equation of all members and differentiating, the inverse kinematics equation of the VGT can be derived as (3).

$$L_k = \left\| \boldsymbol{P}_i - \boldsymbol{P}_j \right\| \tag{2}$$

$$\dot{\boldsymbol{L}} = \mathbf{R}\dot{\boldsymbol{x}} \tag{3}$$

where $\boldsymbol{L} = [L_1, L_2, \cdots, L_{16}]^\top$ is the vector of the link lengths, and matrix $\mathbf{R}$ is the inverse Jacobian of the VGT.

C. Constraints

The proposed VGT has constraints because of the size, power, and collision of mechanical components. The VGT constraints are summarized as follows.

- Length and linear velocity limitation of each member: because of the mechanical design of a linear actuator, the length of the k-th member, L_k, is limited to the range of motion of a linear actuator. In addition, the hardware performance limits the linear actuation speed of the members.

$$L_{min} \leq L_k \leq L_{max} \tag{4}$$

$$\dot{L}_k \leq \dot{L}_{max} \tag{5}$$

- Maintaining minimum manipulability: the VGT should maintain an amount of manipulability to fully control the movement of each member. Therefore, the manipulability of the VGT can be constrained as follows:

$$\kappa = \frac{\sigma_{min}(\mathbf{R})}{\sigma_{max}(\mathbf{R})} \geq \kappa_{min}, \tag{6}$$

  where $\mathbf{R}$ is the inverse kinematics Jacobian in (3). $\sigma_{max}(\mathbf{R})$ and $\sigma_{max}(\mathbf{R})$ are the maximum and minimum singular value of $\mathbf{R}$.
- Ground collision: nodes cannot be below the ground. Therefore, the z component of all nodes should be greater than zero.

$$p_{iz} \geq 0 \tag{7}$$

- Collision between members: the members should not collide with each other. A collision can be prevented by keeping a distance margin between members. We used Lumelsky's distance calculation algorithm for finding this distance [12]. We set a constraint equation to ensure the distance between any two members cannot be less than the margin, to prevent collision. This constraint can be summarized as follows:

$$d_{min}(m_i, m_j) \geq d_0, \tag{8}$$

  where $d_{min}(m_i, m_j)$ is the minimum distance between member i and j.
- Angular collision between adjacent members: the angle between two adjacent members cannot be reduced to less than a certain value, because of the size of the connecting part between a node and member. Therefore,

the angle between the adjacent member is limited to be greater than the collision prevention margin. The angle was calculated using the direction cosine method as follows:

$$\theta_{ijk} = \cos^{-1}\left[\frac{(\boldsymbol{P}_i - \boldsymbol{P}_k)^\top (\boldsymbol{P}_j - \boldsymbol{P}_k)}{\left\|\boldsymbol{P}_i - \boldsymbol{P}_k\right\| \left\|\boldsymbol{P}_j - \boldsymbol{P}_k\right\|}\right] \geq \theta_0. \quad (9)$$

The i-th node and the k-th node in (9) are the adjacent nodes, and the j-th node and the k-th node are also adjacent.

III. Path Planning and Locomotion Algorithm

A. Polygon-based Random Tree Search Algorithm

To determine the desired trajectory to reach the destination point, a RRT search algorithm was modified [9]. PRT identifies a set of desired *Polygon*s by randomly expanding the polygon tree. In the proposed PRT algorithm, the tree structure is as follows:

$$T = \{Polygon, E\} \quad (10)$$

$$Polygon = \{\boldsymbol{Foot}_1, \boldsymbol{Foot}_2, \boldsymbol{Foot}_3\} \quad (11)$$

Nodes in the tree comprise *Polygon*s, which contain *Foot*s that construct the *Polygon*. A *Foot* is a positional vector of one vertex that belongs to a *Polygon*. In the case of VGT, the shape of a support polygon is triangle. Therefore, three *Foot*s are contained on a *Polygon*. The term E in (10) denotes an edge of the tree structure, which describes the relation of parent and child *Polygon*s.

Algorithm 1 shows the overall algorithm for PRT. For each cycle, the objective point of the algorithm is basically randomly selected within the operational region. To ensure the algorithm converges well, the objective point is sometimes set as the goal position with a designated probability, which is denoted as $MixingFactor$ in line 3 of Algorithm 1. The $MixingFactor$ was set as 0.2 during the paper.

The new *Polygon* is expanded from the previous *Polygon* sharing two *Foot*s. Fig. 2 shows the polygon tree expansion schematic. First, the three alternatives of the next *Foot* are placed around the previous *Polygon*. The distances between a *Foot* alternative and two *Foot*s of the previous *Polygon* that are connected to it are set as the same distance, $L_{nominal}$. $L_{nominal}$ represents the initial member length, which was set as 1 in the study. The nearest *Foot* alternative from the objective point, which is randomly determined during the previous step, is selected for setting the next *Foot* of the next *Polygon*. The PRT provides some flexibility to setting the next *Polygon* by permitting a slight distortion within the distortion margin, which is denoted as a circle in Fig. 2. The next *Foot*, which is drawn as a red dot in Fig. 2, is in the closest position within the distortion margin circle. The radius of the distortion margin was set as 0.1 during the study.

The algorithm ends when the last *Polygon* includes the goal position inside of itself. After obtaining the tree from the PRT algorithm, the desired set of *Polygon*s can be found by tracking the parent *Polygon*s from the last *Polygon*. The path planning process is shown in the supplementary video.

Algorithm 1: Polygon-based random tree search

Input: $T = \{Polygon_{init}, E\}, Goal\_Position$

1 $Reach\_Goal = false$;
2 **while** $Reach\_Goal == false$ **do**
3 **if** $Random[0,1] \geq MixingFactor$ **then**
4 $Obj\_Position = Random\_Position$;
5 **else**
6 $Obj\_Position = Goal\_Position$;
7 **end**
8 $Temp\_Foot = FindNearestFoot(Obj\_Position)$;
  `// Making alternative polygon with new foot`
9 $Polygon\_new = MakePolygon(Temp\_Foot)$;
10 **if** $ObstacleCollision(Polygon\_new) == false$ **then**
  `// Add new polygon to the tree`
11 $T.AddPolygon(Polygon\_new)$;
12 $T.AddEdge(Polygon\_pre, Polygon\_new)$;
13 **end**
14 **if** $Goal\_Position \in Polygon\_new$ **then**
15 $Reach\_Goal == true$;
16 **end**
17 **end**
18 **return** T;

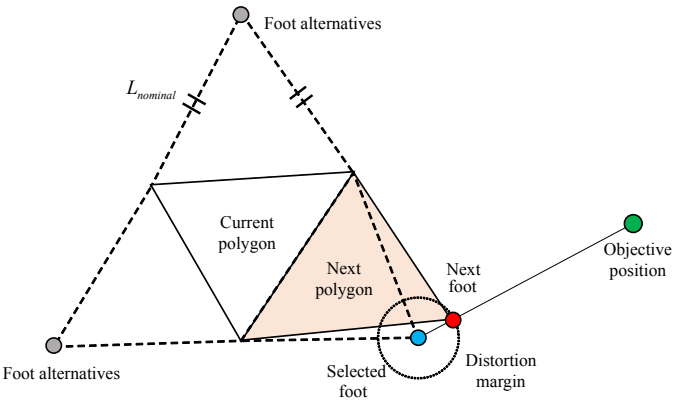


Fig. 2. Schematic diagram of finding next support polygon (*FindNearestFoot* on line 8, Algorithm 1). The closest foot alternative is denoted as a blue dot.

B. Locomotion Algorithm

After the desired set of support polygons were derived using the PRT algorithm, a locomotion algorithm was used to ensure the VGT followed the derived trajectory. The locomotion algorithm was developed from the concept of our previous VGT locomotion algorithm [8]. The proposed locomotion algorithm provides a non-impact rolling motion that prevents dynamic tip-over motion while following the support polygon trajectory. The overall locomotion algorithm is provided in Algorithm 2. The term $\boldsymbol{x}_{CM}$ is the position of the center of mass of the VGT projected on the ground. The term $\dot{\boldsymbol{x}}_{CM,desired}$ is the projected desired velocity vector of the center of mass.

Algorithm 2: Locomotion between two polygons

Input: $Polygon_{(i)}, Polygon_{(i+1)}, \boldsymbol{x}$

1 $\boldsymbol{x}_{CM,init} = Center\_of\_Mass(Polygon_{(i)})$;
2 $Support\_Polygon = Polygon_{(i)}$;
3 $\boldsymbol{C}_i = Center\_of\_Mass(Polygon_{(i)})$;
4 $\boldsymbol{C}_{i+1} = Center\_of\_Mass(Polygon_{(i+1)})$;
5 $\dot{\boldsymbol{x}}_{CM,desired} = MakeCMTrajectory(\boldsymbol{C}_i, \boldsymbol{C}_{i+1})$;
6 **while** $Reach\_Goal == false$ **do**
7 **if** $Stability\_Margin(Support\_Polygon, \boldsymbol{x}_{CM}) \geq 0$ **then**
  `// Moving phase`
8 $\Delta\boldsymbol{x} = MovingControl(\dot{\boldsymbol{x}}_{CM,desired}, \boldsymbol{x}_{CM})$;
9 $\boldsymbol{x}_{new} = \boldsymbol{x}_{pre} + \Delta\boldsymbol{x}$;
10 **else**
  `// Landing phase`
11 $\Delta\boldsymbol{x} = LandingControl(Polygon_{(i+1)}, \boldsymbol{x}_{CM})$;
12 $\boldsymbol{x}_{new} = \boldsymbol{x}_{pre} + \Delta\boldsymbol{x}$;
13 **if** $Foot\_Landing(\boldsymbol{x}_{new}) == true$ **then**
  `// Changing support polygon`
14 $Support\_Polygon = Polygon_{(i+1)}$;
15 **end**
16 **end**
17 **if** $\boldsymbol{x}_{CM} == Center\_of\_Mass(Polygon_{(i+1)})$ **then**
18 $Reach\_Goal = true$;
19 **end**
20 **end**
21 **return** $\boldsymbol{x}_{new}$

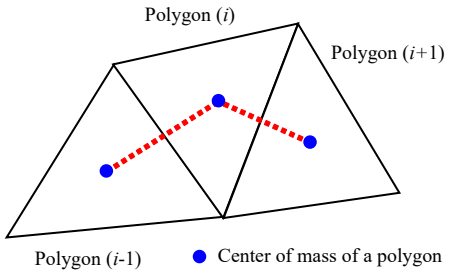


Fig. 3. Finding the projected center of mass trajectory from the support polygon path, which is *MakeCMTrajectory* in Algorithm 2. The center of mass trajectory is denoted as a red dotted line.

The locomotion algorithm ensures the VGT follows both the center of mass and support polygon trajectories. The desired trajectory of the center of mass can be derived from the support polygon trajectory. Fig. 3 shows the projected center of mass trajectory generation process of the locomotion algorithm. By connecting the center of mass point of each *Polygon*s, the desired trajectory for the VGT center of mass is determined. This process is presented from lines 3 to 5 of Algorithm 2.

The locomotion provides a non-impact rolling motion by dividing the motion into two phases, a moving phase and landing phase. During the moving phase, the VGT center of mass is moving through the desired trajectory maintaining the support polygon. The VGT is operated with the moving

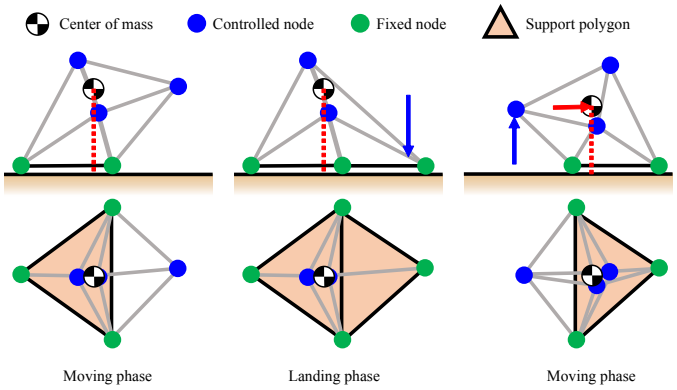


Fig. 4. Non-impact rolling locomotion process.

phase until the projected center of mass is inside the support polygon. When the center of mass is on the border of the support polygon, the VGT is controlled with the landing phase. During the landing phase, the VGT lands the frontal node on the ground without changing the projected position of the center of mass. The frontal node is controlled to land on the vertex of the next desired support polygon. After the frontal node lands on the ground, the support polygon changes, and the VGT is controlled with the moving phase again. Fig. 4 shows the two locomotion algorithm phases.

The velocity of each node $\dot{\boldsymbol{x}}$ is calculated via optimization in each phase. During the moving phase, the node velocity vector is optimized as (12). The optimization function for the moving center of mass was previously provided by Usevitch [7]. With this objective function, the length of each member is near the initial length, ensuring the center of mass follows the desired trajectory. The optimization problem during the moving phase, which is *MovingControl* in Algorithm 2 line 8, can be written as follows:

$$\begin{aligned} \min_{\dot{\boldsymbol{x}}} \ & \left\| \dot{\boldsymbol{x}} + \mathscr{L}\boldsymbol{x} - \boldsymbol{d} \right\| \\ \text{subject to } & \mathbf{M}\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_{CM,desired} \\ & \mathbf{C}_{eq}\dot{\boldsymbol{x}} = 0, \ \mathbf{C}_{ieq}\dot{\boldsymbol{x}} \leq 0 \end{aligned} \tag{12}$$

Here $-\mathscr{L}\boldsymbol{x} + \boldsymbol{d}$ is the optimal $\dot{\boldsymbol{x}}$ to be near to initial member length, where the term $\mathscr{L}$ denotes the graph Laplacian of the VGT and the term $\boldsymbol{d}$ is sum of length and direction of members at each node [7] [13]. The VGT constraints are arranged as $\mathbf{C}_{eq}$ and $\mathbf{C}_{ieq}$, the first is the equality constraint, and the second is inequality constraint.

Similar optimization is used during the landing phase, but with a different objective function and constraints. Equation (13) shows the optimization condition during the landing phase, *LandingControl* in line 11, Algorithm 2. The objective function is selected as the distance between the frontal node and the foot of the next support polygon. During the landing phase, the center of mass movement is restricted by adding the constraint $\mathbf{M}\dot{\boldsymbol{x}} = 0$.

$$\begin{aligned} \min_{\dot{\boldsymbol{x}}} \ & \left\| \boldsymbol{P}_{front} - \boldsymbol{F}_{next} \right\| \\ \text{subject to } & \mathbf{M}\dot{\boldsymbol{x}} = 0 \\ & \mathbf{C}_{eq}\dot{\boldsymbol{x}} = 0, \ \mathbf{C}_{ieq}\dot{\boldsymbol{x}} \leq 0. \end{aligned} \tag{13}$$

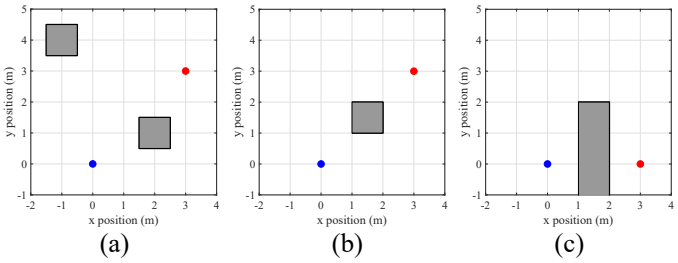


Fig. 5. Three simulation cases that have different obstacles. The starting point is indicated as a blue dot and the goal point is indicated as a red dot. (a) Two obstacles; (b) One obstacle in the middle; (c) Long one obstacle.

$\boldsymbol{P}_{front}$ denotes the positional vector of the frontal node and $\boldsymbol{F}_{next}$ is the objective foot positional vector of the next support polygon.

IV. Simulation Results

The advantages of the proposed PRT algorithm were proved via MATLAB simulation. Two path planning algorithms were compared. The first was our proposed PRT algorithm and the second was the original RRT algorithm for the center of mass path planning, coupled with our previous algorithm for locomotion [8] to follow the desired center of mass trajectory. The step size of the original RRT algorithm was set as 0.1 m. The parameters, including initial link lengths and constraints are listed in Table I. The parameters were determined in such a way that both algorithms have the possibility of succeeding. L_{long} in Table I is the length of the long member in the initial state, the exterior members of the octahedral shape. L_{short} is the length of the interior members. The other terms were defined in the previous section, from (4) to (9).

TABLE I
Simulation Parameters

	L_{init}	L_{min}	L_{max}	$\dot{L}_{max}$	κ_{min}	d_0	θ_0
L_{long}	1.0 m	0.5 m	2.0 m	0.015 m/s	0.10	0.10 m	10°
L_{short}	0.71 m	0.3 m	1.8 m				

Three environments were selected for the locomotion simulation. The first case is an environment with two square obstacles. The second case is an environment with one obstacle in the middle. Finally, the third case features a long rectangular obstacle that blocks the bottom way to the goal position. The three cases were presented on Fig. 5. The work space is a rectangle with range (-1 m, 5 m) in both directions x and y.

Path planning and locomotion with the two algorithms was repeated 20 times for each case, because the two algorithm have a random component. The RRT algorithm succeeded in producing a valid path that satisfies the constraints only in case 1. Therefore, the two algorithms can only be compared in case 1. The simulation results on the case 1 are summarized in Table II. All data in Table II are averages of all successful paths. The PRT success rate is superior to that

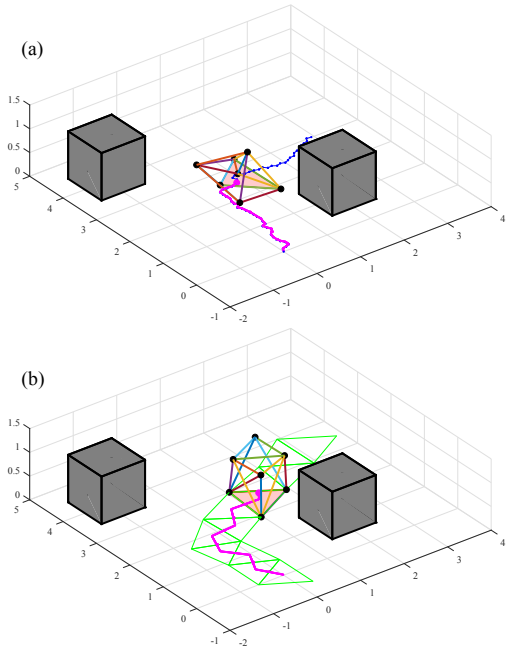


Fig. 6. VGT movement using two different algorithms on case 1. The center of mass trajectory is denoted as a purple line. (a) Original RRT combined with the previous locomotion algorithm. (b) Proposed PRT algorithm.

of the RRT via reducing VGT distortion during locomotion. Because of constraint parameter adjusting for comparison, the success rate of PRT was reduced with respect to its maximum capability. Furthermore, the mean PRT stability margin is much greater than that of the RRT by maintaining the shape of support polygons, as seen in the supplementary video.

TABLE II
Comparison between two algorithm on case 1

	Success rate	Stability margin	Planning time		Locomotion time
			Path	Locomotion	
RRT	10%	0.094 m	0.020 s	1839.5 s	124.7 s
PRT	55%	0.121 m	0.535 s	182.3 s	134.5 s

The computational time of all planning process (path and locomotion planning) was much smaller when using the PRT algorithm as shown in Table II. Although path planning time of the PRT is about 0.5 s longer than that of the RRT, locomotion planning time of the PRT is 10 times shorter than that of the RRT. This is because large distortion of the VGT makes it difficult for the algorithm to find rolling motion without violating constraints. Locomotion time, the arrival time at the goal point, is about 10 s longer on the PRT. However, the RRT showed very low success rate and long locomotion planning time, which makes this small advantage of the RRT negligible.

Fig. 6 shows the locomotion simulation using the two different algorithms. When using the original RRT algorithm, the VGT center of mass followed the desired trajectory. As

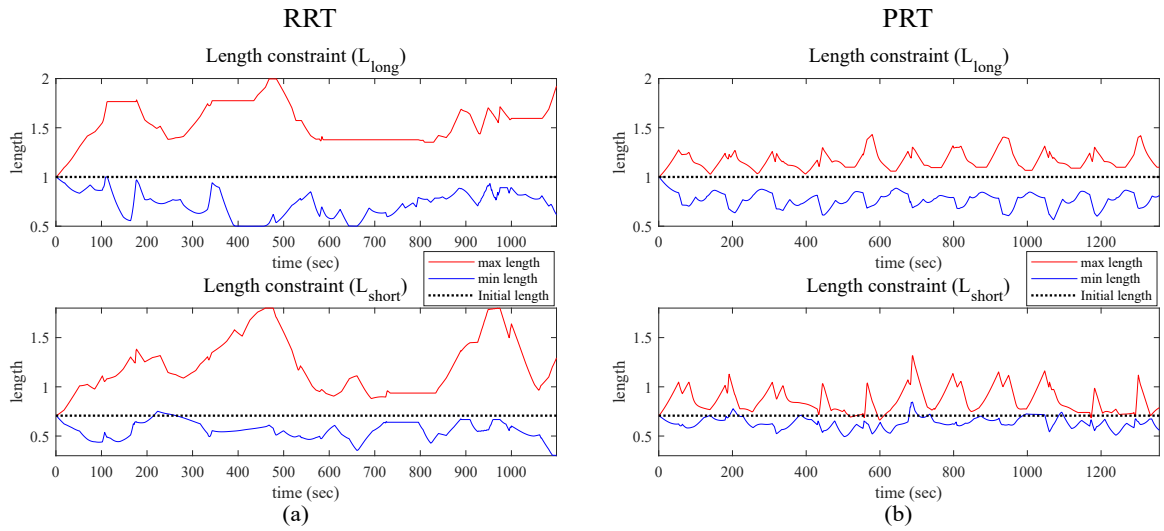


Fig. 7. Maximum and minimum members lengths with respect to simulation time. L_{long} is the long member, surrounding the octahedron, and L_{short} is the short member inside the octahedron. The red line is the maximum length within all members and the blue line is the minimum length. (a) Results using the RRT, L_{long} variation 0.46 m, L_{short} variation 0.45 m; (b) results using the PRT, L_{long} variation 0.22 m, L_{short} variation 0.16 m.

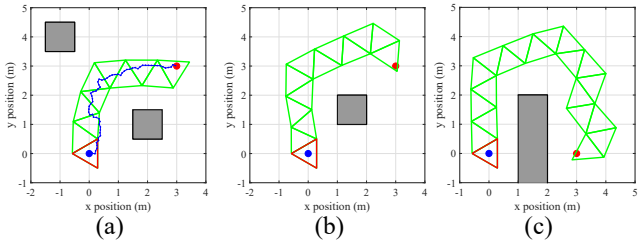


Fig. 8. Desired paths derived using the two algorithms for each simulation case. The initial VGT support polygon is denoted as a red triangle. The desired support polygon trajectory from PRT is denoted as a green line. The desired center of mass path derived from the original RRT is denoted as a blue line. The red dot is the goal position and obstacles are presented as gray squares. (a) Case 1; (b) Case 2; (c) Case 3.

shown in Fig. 6 (a), VGT distortion is large during locomotion. In contrast, VGT distortion is significantly reduced when using the proposed PRT algorithm as shown in Fig. 6 (b). In the PRT algorithm, large VGT distortion is prevented by designating support polygons. Locomotion simulations of two algorithms are presented in a supplementary video of the paper. The amount of distortion can be estimated by examining the length of each member. The amount of distortion increases when the difference between the current and initial lengths of each member increases. Fig. 7 shows the maximum and the minimum lengths of members and their initial length. Using the PRT, the member lengths did not extend beyond a certain region, while the member lengths are near the constraint boundary using the original RRT. Therefore, it can be concluded that using the PRT significantly reduces VGT distortion during locomotion.

In addition, the PRT produced suitable paths for the case 2 and case 3 that satisfy system constraints. The successful paths of three cases with two algorithms are drawn in Fig. 8. The RRT path is drawn only for case 1, because the RRT only succeed in case 1. The results of all cases with the PRT are summarized on Table III. The success rates of case 2 and 3 were smaller than the ones of case 1. The planning and locomotion time was increased from case 1 to case 3 because of the complexity of the derived paths. Therefore, the proposed PRT algorithm can produce suitable paths on various environments with decent success rate.

TABLE III

SIMULATION RESULT OF THE PRT ON ALL CASES

	Success rate	Stability margin	Planning time		Locomotion time
			Path	Locomotion	
Case 1	55%	0.121 m	0.535 s	182.3 s	134.5 s
Case 2	30%	0.121 m	0.173 s	230.8 s	172.1 s
Case 3	30%	0.122 m	0.123 s	359.1 s	248.1 s

V. CONCLUSIONS

In this paper, a path planning algorithm for VGT based on a support polygon, PRT, was presented. PRT provides stable path planning and locomotion on various environments with efficient constraint clearances. PRT was compared to the original RRT algorithm with path planning and locomotion simulation in three environments with obstacles. PRT not only showed a higher success rate compared to that of the original RRT but also provided stable locomotion. Furthermore, the PRT succeeded to produce a path for two cases, which couldn't be solved by the previous RRT. The PRT algorithm will be improved for application to uneven surfaces of various slopes. Also, the possibility for applying learning algorithms to the PRT will be researched.

ACKNOWLEDGMENT

We thank to Seongjae Jeong, Jinkyu Kim, Byungheon Kim, Kyumin Park, Seunghyun Kim and Tae Gyun Ahn for providing RRT program code.

REFERENCES

[1] 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.

[2] P. C. Hughes, W. G. Sincarsin, and K. A. Carroll, "Trussarm-a variable-geometry-truss manipulator," *Journal of Intelligent Material Systems and Structures*, vol. 2, no. 2, pp. 148–160, 1991.

[3] G. J. Hamlin and A. C. Sanderson, "Tetrobot modular robotics: Prototype and experiments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, vol. 2. IEEE, 1996, pp. 390–395.

[4] S. Curtis, M. Brandt, G. Bowers, G. Brown, C. Cheung, C. Cooperider, M. Desch, N. Desch, J. Dorband, K. Gregory, *et al.*, "Tetrahedral robotics for space exploration," in *2007 IEEE Aerospace Conference*. IEEE, 2007, pp. 1–9.

[5] A. Spinos, D. Carroll, T. Kientz, and M. Yim, "Variable topology truss: Design and analysis," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2717–2722.

[6] W. H. Lee and A. C. Sanderson, "Dynamic rolling locomotion and control of modular robots," *IEEE Transactions on robotics and automation*, vol. 18, no. 1, pp. 32–41, 2002.

[7] N. Usevitch, Z. Hammond, S. Follmer, and M. Schwager, "Linear actuator robots: Differential kinematics, controllability, and algorithms for locomotion and shape morphing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5361–5367.

[8] S. Park, E. Park, M. Yim, J. Kim, and T. Seo, "Optimization-based nonimpact rolling locomotion of a variable geometry truss," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 747–752, 2019.

[9] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[10] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3. IEEE, 2002, pp. 2383–2388.

[11] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 723–730.

[12] V. J. Lumelsky, "On fast computation of distance between line segments," *Information Processing Letters*, vol. 21, no. 2, pp. 55–61, 1985.

[13] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *International Journal of control*, vol. 82, no. 3, pp. 423–439, 2009.