2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

Locomotion Planning of a Truss Robot on Irregular Terrain

Jangho Bae[1], Inha Park[2], Mark Yim[1], and TaeWon Seo[2], *Senior Member, IEEE,*

***Abstract*— This paper proposes a new locomotion algorithm for truss robots on irregular terrain, in particular, for the Variable Topology Truss (VTT) system. The previous Polygon-based Random Tree (PRT) search algorithm for support polygon generation is extended to irregular terrain while considering friction and internal force limitations. By characterizing terrain, unreachable areas are excluded from search to increase efficiency. A one-step rolling motion primitive is generated based on the kinematics, statics, and constraints of VTT. The locomotion planning is completed by transforming and connecting multiple motion primitives with respect to the desired support polygons. The algorithm's performance is verified by conducting simulations in multiple types of environments.**

I. Introduction

Variable Geometry Truss (VGT) is a modular truss robot system that comprises linearly actuated truss members and passive spherical joints. Since Miura et al. introduced the VGT concept and proposed possible applications, the VGT concept is applied to mobile systems for exploration [1]–[4]. We are developing the Variable Topology Truss (VTT) system [5]. The VTT system can change its topology, which is not possible for previous VGTs. The main application of VTT is to perform search and rescue operations in disaster sites, which includes finding victims and shoring debris [6]. Fig. 1 shows the hardware prototype and conceptual demonstration of VTT. We chose the octahedral configuration for locomotion studies because the shape is symmetric and it can perform impact-free rolling. Although this work is based on the kinematics and constraints of an octahedral VTT, the algorithm can be adapted for arbitrary truss robots with triangular faces.

Locomotion is an essential task for performing search and rescue operations. In a locomotion task for a polyhedral robot, the task is to move the robot from an initial point to some goal point, typically a long distance compared to the overall size of the robot while maintaining all of the physical constraints. There are several approaches to this problem. One method is to apply a continuous optimization solver that greedily moves the robot towards the goal [7], [8]. Other authors have investigated reinforcement learning techniques

*This research was supported by the MOTIE (Ministry of Trade, Industry, and Energy) in Korea, under the Human Resource Development Program for Industrial Innovation(Global) (P0017306, Global Human Resource Development for Innovative Design in Robot and Engineering) supervised by the Korea Institute for Advancement of Technology (KIAT). *(Co-first author: Jangho Bae and Inha Park) (Corresponding author: TaeWon Seo)*

[1]J. Bae and M. Yim are with the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104 USA jangho.bae91@gmail.com; yim@seas.upenn.edu

[2]I. Park and T. Seo are with the Department of Mechanical Engineering, Hanyang University, Seoul 04763, Republic of Korea dlsgk990@hanyang.ac.kr; taewonseo@hanyang.ac.kr

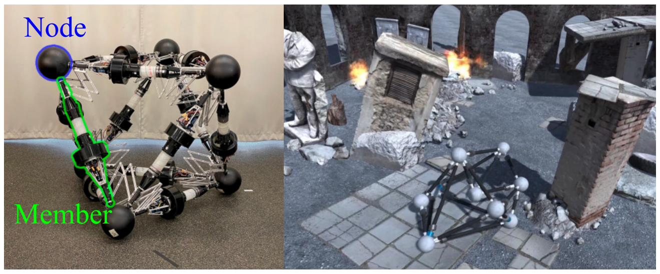

Fig. 1. The VTT introduction figure: (a) Hardware prototype of octahedral VTT; (b) Computer generated VTT demonstration graphics in a disaster site.

[9]. Another method is to break up the problem into two parts: first generating a sequence of support polygons and then generating the motions required to make each step. This approach can use a much more efficient planner to plan the overall trajectory, which breaks the motion into a series of easy-to-solve steps.

We previously proposed a Polygon-based Random Tree (PRT) search algorithm to generate support polygon sequence [10]. The PRT algorithm generates a desired set of support polygons to reach a goal based on the Rapidly-exploring Random Tree (RRT) search [11]. PRT can prevent large distortion of VTT by limiting the shape of support polygons. The previous algorithms can only generate support polygons on a flat surface. However, the locomotion required for search and rescue operations should target more complex environments. An expansion to irregular terrain is required to achieve VTT's objectives.

Terrain characterization can make probabilistic planners like PRT more efficient. Certain parts of the terrain may be inaccessible due to a steep incline angle or sudden changes in surfaces. Excluding these parts of the terrain from the search area can reduce the amount of calculation and time for path planning. The study of Brunner et al. presented locomotion planning of a tracked vehicle by combining terrain characterization and Dijkstra search [12]. Saboia et al. introduced the mobile manipulator and corresponding algorithm that determines reachable area [13].

Our previous non-impact rolling algorithm uses an optimization solver to find a smooth node trajectory without tipping over [10], [14]. However, the algorithm is slow and sometimes cannot find a solution. Another approach is to use a motion primitive, which is a pre-calculated optimal motion for a specific situation. Usevitch et al. presented locomotion planning of VGT systems with the motion primitive concept [15]. However, the study only considered regular support polygons and flat terrain.

This paper proposes the locomotion planning strategy on irregular terrain for a truss robot. We chose our VTT system as an example for the algorithm. We expanded the PRT algorithm to be able to generate support polygon sequences on irregular terrain while considering friction and internal forces. Simple terrain characterization is applied to exclude unreachable regions from given terrains. We used a motion primitive concept to make the node trajectory planner more reliable and faster. A one-step motion on flat ground is generated and transformed to make it fit into the support polygon sequence. The locomotion planning strategy is verified by simulations in various environments.

II. Variable Topology Truss Analysis

A. Kinematic Analysis

As stated in the introduction section, we selected an octahedron topology for locomotion. The spherical joints are called *Nodes* and the trusses are called *Members*, as denoted in Fig. 1. The length of the members are merged into a vector $\boldsymbol{L} = [L_1, \cdots, L_{12}]^\top$, and the initial length is called the nominal length L_{nom}. The position of node i is indicated as $\boldsymbol{P}_i$ and all positions are merged into one vector $\boldsymbol{x}$ as follows:

$$\boldsymbol{x} = [p_{1x}, \cdots, p_{6x}, p_{1y}, \cdots, p_{6y}, p_{1z}, \cdots, p_{6z}]^\top. \tag{1}$$

Two kinds of kinematic analysis are conducted for VTT. First is the center of mass differential kinematics analysis. The relation between the projected velocity of the center of mass, $\dot{\boldsymbol{x}}_C = [\dot{x}_{Cx}, \dot{x}_{Cy}]^\top$, and node velocity vector $\dot{\boldsymbol{x}}$ can be calculated as follows:

$$\dot{\boldsymbol{x}}_C = \mathbf{M}\dot{\boldsymbol{x}}, \tag{2}$$

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

Second is the inverse kinematics analysis. The inverse kinematics corresponds to the calculation of member lengths given the node positions. Differential inverse kinematics can be derived by calculating the member length vector $\boldsymbol{L}$ with respect to the node position vector $\boldsymbol{x}$, then differentiating with respect to time as follows:

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

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

B. Static Analysis

Static analysis is critical for locomotion on irregular terrain. VTT uses spiral zipper actuators for the linear actuation of truss members, and the actuator has force limits that need to be considered [16]. The static analysis includes the internal force experienced by the members and reaction forces applied to the ground-contacting nodes, *ground nodes*.

For analysis, the force and moment equilibrium equations were derived for the members and the nodes. Free-body diagrams of a member and a ground node are presented in Fig. 2. $\boldsymbol{F}_{n,1}$ and $\boldsymbol{F}_{n,2}$ are forces applied from nodes to the member, and $\boldsymbol{w}_n$ is the weight of the member. $\boldsymbol{L}_n$ is a vector from $\boldsymbol{P}_i$ to $\boldsymbol{P}_j$, and $\boldsymbol{L}_{CM}$ is a vector from $\boldsymbol{P}_i$ to the center of mass of the member. $\boldsymbol{F}_{R,i}$ is the reaction force from the

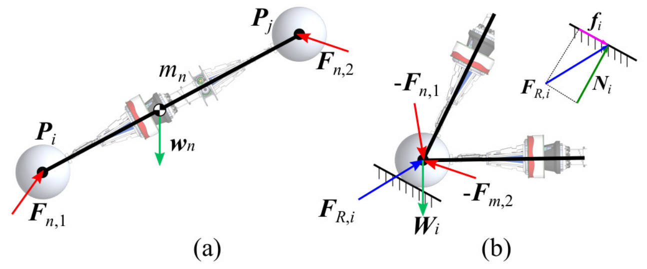

Fig. 2. Static analysis diagrams. (a) Free-body diagram of the n-th member that connects node i and j; (b) Free-body diagram of ground node i that connects member m_n and m_m.

ground, and $\Sigma \boldsymbol{F}_{member}$ denotes the sum of the forces from all connected members. If the node is not a ground node, $\boldsymbol{F}_{R,i}$ is zero. $\boldsymbol{W}_i$ indicates the weight of the node.

VTT on an irregular terrain is an indeterminate structure. Therefore, an optimization solver is required to solve the static problem. We calculate the vector $\boldsymbol{X}$ as follows:

$$\min \left\|\boldsymbol{X}\right\| \text{ subject to } \mathbf{A}_f\boldsymbol{X} = \boldsymbol{B}_f,\ \left\|\boldsymbol{f}_i\right\| \leq \mu \left\|\boldsymbol{N}_i\right\|, \tag{4}$$

where $\boldsymbol{X} = [\boldsymbol{F}_{1,1}, \boldsymbol{F}_{1,2}, \cdots, \boldsymbol{F}_{12,2}, \boldsymbol{F}_{R,1}, \cdots, \boldsymbol{F}_{R,i}]^\top$ is a merged force vector. All equilibrium equations are merged into matrix form as $\mathbf{A}_f\boldsymbol{X} = \boldsymbol{B}_f$. The second constraint indicates that the required friction $\boldsymbol{f}_i$ should be smaller than maximum static friction, where μ is the maximum static friction coefficient and $\boldsymbol{N}_i$ is the normal force. The compression or tensile force applied to n-th member is calculated from $\boldsymbol{F}_{n,1}$ and $\boldsymbol{F}_{n,2}$ [17].

C. Constraints

The constraints for simulation are chosen to match the capabilities of the current VTT hardware. Most constraints are determined by the performance of the spiral zipper actuators. The constraints considered in this study are: *member length limits*, *angle limits between adjacent members*, *member force limits*, *manipulability lower limit*, *collision avoidance*, and *maximum required friction*. *Member force limits* constrain the maximum compression or tensile force applied to each member. *Maximum required friction* constraint prevents the required friction from exceeding the maximum static friction force, which is presented in Section II B. The constraints are presented as the function $\boldsymbol{f}_{con}(\boldsymbol{x})$ in the rest of the paper.

III. Path Planning and Locomotion Algorithm

A. Motion Primitive Generation

We generated a one-step motion between two regular support polygons on flat ground and used the motion as a motion primitive. The motion primitive comprises four phases: moving, landing, transient, and returning. A brief introduction for each phase is presented in Fig. 3. Ground nodes are fixed on the ground, and only controlled nodes are allowed to move. The details of each phase are presented in the previous paper [10], [14], [18].

In the moving phase, VTT is controlled to move its center of mass along the line between two centers of support polygons. The moving phase is maintained until the distance from the center of mass to the support polygon edge is

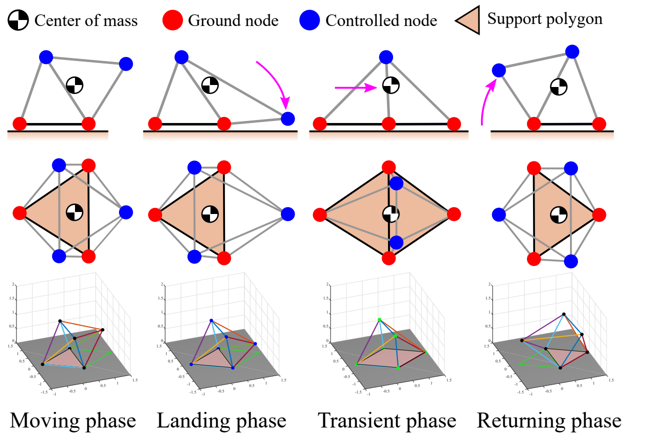

Fig. 3. Four phases of the motion primitive. Only controlled nodes are allowed to move. The bottom four figures are snapshots from calculated motion primitive.

smaller than the stability margin. During the moving phase, the movement of VTT nodes is obtained by minimizing the following objective function:

$$\min_{\dot{\boldsymbol{x}}} \left\|\dot{\boldsymbol{x}}+\mathscr{L}\boldsymbol{x}-\boldsymbol{d}\right\| \tag{5}$$

where $\mathscr{L}$ is the graph Laplacian and the term $\boldsymbol{d}$ is sum of the length and direction of members at each node [8]. The objective function is designed to find the next movement $\dot{\boldsymbol{x}}$ to maintain member length as close as possible to its initial length.

VTT stops moving its center of mass and lands the front node on the ground during the landing phase. Once the front node is landed on the ground, the landing phase ends. The landing phase movement is derived by finding the node trajectory that minimizes the distance between the front node and the landing position. During the transient phase, the center of mass is moved into the next support polygon while four nodes are contacting the ground. The transient phase movement is calculated from the same objective function for the moving phase, but with different ground nodes. The transient phase ends when the center of mass moves to the next support polygon and has a distance larger than the stability margin to the support polygon edge. The returning phase is designed to make VTT return to its initial pose in the motion primitive. The initial and final configuration of the motion primitive are the same to allow chaining this one motion primitive repeatedly. The sequence of $\boldsymbol{x}$ calculated by the phases forms the motion primitive $u(t)$, where $u(0)$ is the initial pose $\boldsymbol{x}_{ini}$, and $u(t_{fin})$ is the final pose $\boldsymbol{x}_{fin}$.

B. Terrain Characterization

Unreachable areas of the given target terrain are excluded from the search area of the PRT algorithm by characterizing the terrain. We selected the incline angle of surfaces as a criterion. An object can stand on the surface without slipping when the following condition is satisfied.

$$\alpha \leq \tan^{-1}\mu, \tag{6}$$

where α is the incline angle of the surface and μ is the maximum static friction coefficient of the terrain. However,

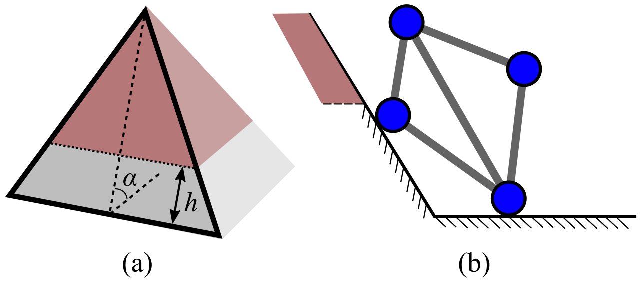

Fig. 4. Terrain exclusion algorithm. The excluded area is colored as grey; (a) Excluded part of a terrain mesh that has an incline angle larger than slip condition; (b) VTT holding its position by stepping partially on a high-inclined surface.

if VTT is partially on a surface that has a larger incline angle, VTT may be able to stand without slipping depending on the situation. The situation is shown in Fig. 4 (b). Therefore, we allow a certain portion of the steep surface to include the possibility. In summary, the ground surfaces that have a higher incline angle than the slip condition are excluded from the search area, except for a certain portion near the base ground. For example, a friction coefficient $\mu = 0.5$ leads to a max angle 26.6°. The terrain characterization method is denoted in Fig. 4 (a). The height of a regular, nominal-sized support polygon is denoted as h, which is the same as $(\sqrt{3}/2)L_{nom}$.

C. PRT Algorithm for Irregular Terrain

The PRT algorithm is expanded to find desired support polygons on irregular terrain. This is summarized in Algorithm 1. The PRT algorithm uses a tree structure to find support polygons to reach the goal position. [10] The tree T comprises *Polygons* and connections between *Polygons*. Each *Polygon* represents a possible support polygon. One *Polygon* includes three *Foots*, which are vertices of the *Polygon*, as follows:

$$Polygon = \{\boldsymbol{Foot}_1, \boldsymbol{Foot}_2, \boldsymbol{Foot}_3\}, \tag{7}$$

where $\boldsymbol{Foot}_i$ is a position of the i-th vertex of the *Polygon*.

The objective point, *Obj_Point*, is set randomly within the workspace for every iteration. The objective point is occasionally located at the goal position with the possibility of the mixing factor, which is set as 0.1 for this study.

Temporary *Foots* are generated from every existing *Polygon* for every iteration. The function *FootGen* in Algorithm 1 denotes the foot generation process. *FootGen* finds points that contact the ground surface for each direction except toward the *Polygon*'s parent. The distances from two vertices to the corresponding temporary *Foot* are the same as the nominal length, L_{nom}. Fig. 5 shows the *Foot* generation method for one *Polygon*. All generated candidate *Foots* are stored in $Foot_{set}$.

The nearest *Foot* from the objective point is set as the next Foot candidate, $\boldsymbol{Foot}_{new}$. After that, $\boldsymbol{Foot}_{new}$ is moved a small distance towards the objective point. This distance is called a distortion margin, which gives flexibility to the shape of support polygons. $\boldsymbol{Foot}_{new}$ determination process is denoted as *FindNextFoot* in Algorithm 1.

Algorithm 1: Polygon-based Random Tree search for irregular terrain

```
Input: $T = InitTree(Polygon_{init}), Goal$
1  $Reach\_Goal = False$;
2  while $Reach\_Goal == False$ do
3      if $Random[0,1] \geq MixingFactor$ then
4          $Obj\_Point = Random\_Position()$;
5      else
6          $Obj\_Point = Goal$;
7      $Foot_{set} = FootGen(T)$;
8      $\boldsymbol{Foot}_{new} = FindNextFoot(Obj\_Point, Foot_{set})$;
9      if $\boldsymbol{Foot}_{new} \in ExcludedGround$ then
10         continue;
11     $Polygon_{new} = MakePolygon(\boldsymbol{Foot}_{new})$;
12     $\boldsymbol{Pose}_{new} = FindPose(Polygon_{new})$;
13     if $ConsViolation(\boldsymbol{Pose}_{new}) == False$ then
14         $T.AddPolygon(Polygon_{new})$;
15     if $Goal \in Polygon_{new}$ then
16         $Reach\_Goal = True$;
17 $S = T.ExtractSequence()$;
18 return $S$;
```

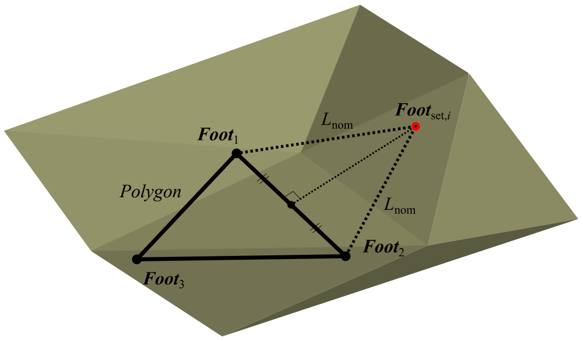

Fig. 5. *Foot* generation method for PRT in irregular terrain. The red dot indicate the generated i-th candidate, $\boldsymbol{Foot}_{set,i}$.

A nominal pose, $\boldsymbol{Pose}$, is the desired pose of VTT to stand on the corresponding *Polygon*. For nominal pose calculation, the *Polygon* is projected to the xy plane The nominal pose is calculated by minimizing the difference between the member length and the nominal length L_{nom} on the projected *Polygon*. After optimization, the ground nodes are moved back to the original *Polygon*. If $\boldsymbol{Pose}_{new}$ satisfies all the constraints, $Polygon_{new}$ is added in the tree T. Nominal poses are used to transform the motion primitive during the later process.

Once $Polygon_{new}$ includes the *Goal*, the PRT algorithm is terminated. From the final *Polygon*, the algorithm tracks through the connection data until it finds the first *Polygon*. The *Polygon* sequence from the first to the final is returned as S.

D. Node Trajectory Planning

The node trajectory for following the desired support polygons is calculated by transforming and connecting the motion primitive u for each polygon. Algorithm 2 presents the method of locomotion planning from the motion primitive. This method is inspired by Hauser's study about humanoid robots [19]. First, the function *MatchNode* in Algorithm 2 swaps the motion primitive's node numbers to match the support polygon's ground nodes. The node-matched motion primitive is written as $\hat{u}$.

Algorithm 2: Node trajectory generation based on the motion primitive

```
Input: $u, S$
1 for $i = 1$ to $length(S) - 1$ do
2     $\hat{u} = u.MatchNode(S_i.Node, S_i.Polygon)$;
3     $\tilde{u} = \hat{u}.Transform(S_i.Pose, S_{i+1}.Pose)$;
4     while $ConsViolation(\tilde{u}) == True$ do
5         $\tilde{u}.Revision$;
6     $Trajectory.Add(\tilde{u})$;
7 return $Trajectory$;
```

Next, the linear transformation is found to match the initial and final pose of the motion primitive with the nominal pose of i-th and $(i+1)$-th *Polygons*. The nominal pose of the i-th Polygon is denoted as $S_i.Pose$. The linear transformation from $\hat{u}$ to $\tilde{u}$ can be calculated as follows:

$$\tilde{u}(t) = \mathbf{A}(\hat{u}(t) - \hat{\boldsymbol{x}}_{ini}) + S_i.Pose, \tag{8}$$

where $\tilde{u}$ is the transformed motion primitive that has $S_i.Pose$ as its initial pose and $S_{i+1}.Pose$ as its final pose. $\hat{\boldsymbol{x}}_{ini}$ is the initial node position of the motion primitive after *MatchNode*. The linear transform matrix $\mathbf{A}$ is calculated separately for each node position and then merged into one matrix. The partial matrix $\mathbf{A}_n$, the transform for the n-th node, can be calculated as follows:

$$\mathbf{A}_n = \mathbf{I}_{3\times3} + \frac{(\Delta\tilde{\boldsymbol{P}}_n - \Delta\boldsymbol{P}_n)\Delta\boldsymbol{P}_n^\top}{\left\|\Delta\boldsymbol{P}_n\right\|^2}, \tag{9}$$

where $\Delta\tilde{\boldsymbol{P}}_n$ is a difference between the n-th node position in $S_i.Pose$ and $S_{i+1}.Pose$. $\Delta\boldsymbol{P}_n$ is the difference between the initial and the final position of $\hat{u}$. The partial matrices $\mathbf{A}_n$ are selected to be as close as possible to the identity matrix [19]. This process is presented in Fig. 6 (a).

Every time frame of the linearly transformed motion primitive $\tilde{u}$ is checked for constraint violations. If there are any constraint violations, $\tilde{u}$ is transformed again to avoid constraint violations. First, the algorithm finds the revision point, $\boldsymbol{x}_{rev}$, which is the point that has the largest constraint violation amount. The size of constraint violation is defined as the norm of the vector of all violated $\boldsymbol{f}_{con}(\boldsymbol{x})$ values. $\boldsymbol{x}_{rev}$ is revised as follows:

$$\tilde{\boldsymbol{x}}_{rev} = \boldsymbol{x}_{rev} - k(\mathbf{I} - \mathbf{C}_{fix})\sum_i \nabla f_{con,i}(\boldsymbol{x}_{rev}), \tag{10}$$

where $\nabla f_{con,i}$ is the gradient of the i-th violated $\boldsymbol{f}_{con}$ with respect to $\boldsymbol{x}$. The matrix $\mathbf{I} - \mathbf{C}_{fix}$ selects the components of the controlled node. k is an arbitrary constant that adjusts the

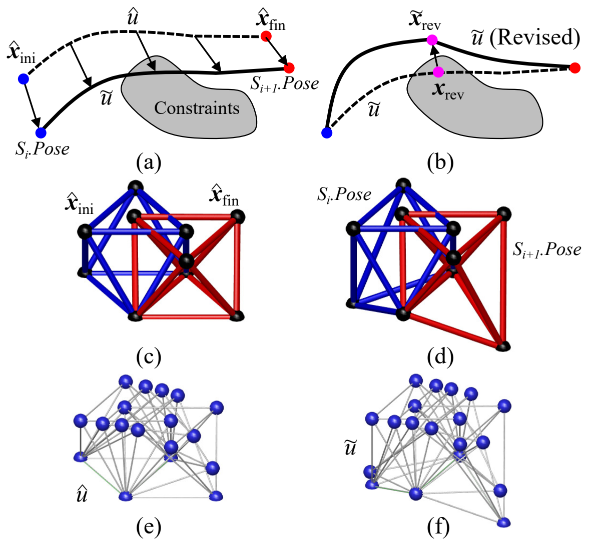

Fig. 6. Transformation of the motion primitive; (a) Linear transformation of $\hat{u}$ to match the initial and final pose; (b) Revision for avoiding constraint violation; (c) Initial and final pose of the motion primitive; (d) Initial and final pose of the transformed motion primitive; (e) The motion primitive $\hat{u}$; (f) Transformed motion primitive $\tilde{u}$.

amount of revision. The equation (10) moves $\boldsymbol{x}_{rev}$ outside of the constraint violation region. Calculation of $\tilde{\boldsymbol{x}}_{rev}$ is iterated until the algorithm finds $\tilde{\boldsymbol{x}}_{rev}$ that satisfies the constraints.

After finding $\tilde{\boldsymbol{x}}_{rev}$, the prior and the posterior parts of $\tilde{u}$ are linearly transformed with the same method above. Fig. 6 (b) describes the revision of $\tilde{u}$ for avoiding constraint violations. The linear transformation of the prior and the posterior parts are calculated separately from applying equations (8) and (9). The constraints violation of the revised $\tilde{u}$ is rechecked after revision. This process *Revision* is iterated until the revised $\tilde{u}$ does not violate the constraints. Then the revised $\tilde{u}$ is added to the locomotion trajectory after the process. Algorithm 2 is terminated after generating the node trajectory for every *Polygon* in S.

IV. Simulation Results and Discussions

Simulations were conducted to prove the performance of locomotion planning on irregular terrain. The constraint values are set by considering the current VTT hardware as presented in Table I. L_{min} and L_{max} are the members' length limits, while θ_{min} and θ_{max} are limits of adjacent member angles. κ is the minimum manipulability limit. The internal force limits for every member are denoted as F_{comp} for compressive force and F_{tens} for tensile force.

TABLE I
Constraints values

L_{min}	L_{max}	θ_{min}	θ_{max}	κ	F_{comp}	F_{tens}	μ
1.1 m	3.25 m	25°	155°	0.2	200 N	100 N	0.5

The simulations were conducted in three environments, which are inspired by Hirose's study [20]. The first environment is a terrain with two types of holes, large and small. Each hole's size, location, and depth were randomly generated within a certain range. The second environment is a terrain with multiple spikes and holes. Similar to the first environment, each component's dimensions were randomly generated. The last environment is a hill terrain with various randomized incline angles. The empty space of every environment was filled with rough terrain.

Fig. 7 shows the support polygon sequences and locomotion simulation results for three environments. The blue polygons are generated support polygons in the tree T, while green polygons are the desired support polygons from the initial point to the goal in S. The excluded surfaces by terrain characterization were colored red. The support polygon generation processes and VTT movements are presented in the supplementary video.

We conducted locomotion planning for each environment, with and without terrain characterization. Each case was repeated ten times to eliminate the randomness of the algorithm. We checked the calculation time of support polygon planning (PRT) and Motion Primitive Transformation (MPT). The calculations were conducted by a laptop computer (Intel Core i7 Processor, 6 cores, 2.60 GHz, 32 GB RAM). The mean values of calculation times are organized in Table II, where TC stands for Terrain Characterization.

TABLE II
Calculation time results

Time	Environment 1		Environment 2		Environment 3	
(sec)	PRT	MPT	PRT	MPT	PRT	MPT
w/ TC	325.7	1113.4	366.7	950.3	281.8	666.7
w/o TC	418.4	1211.0	388.6	972.8	537.4	758.4

The results show that applying terrain characterization lowers the PRT calculation time by reducing the number of random searches. Terrain characterization prevents meaningless searches on unreachable surfaces. Therefore, the effect of terrain characterization increases proportionally to the area of unreachable surfaces. Since environment 3 has the largest unreachable area, the difference in calculation time is the largest among the three environments.

The terrain characterization also decreases the calculation time of MPT, but not as much as it reduces the PRT calculation time. The number of support polygons for each result was similar. Since the MPT calculation time is roughly proportional to the number of support polygons, it is natural that terrain characterization gives less effect on MPT planning time. The small improvement comes from decreased revision number of the motion primitive. Without terrain characterization, the support polygon is more likely to be placed on a badly conditioned surface. It may increase the number of motion primitive revisions because a badly conditioned surface may cause more severe constraint violations.

The MPT revision iteration process was successfully terminated in most cases. The iteration may not end if severe constraint violation occurs. Since severe constraint violations

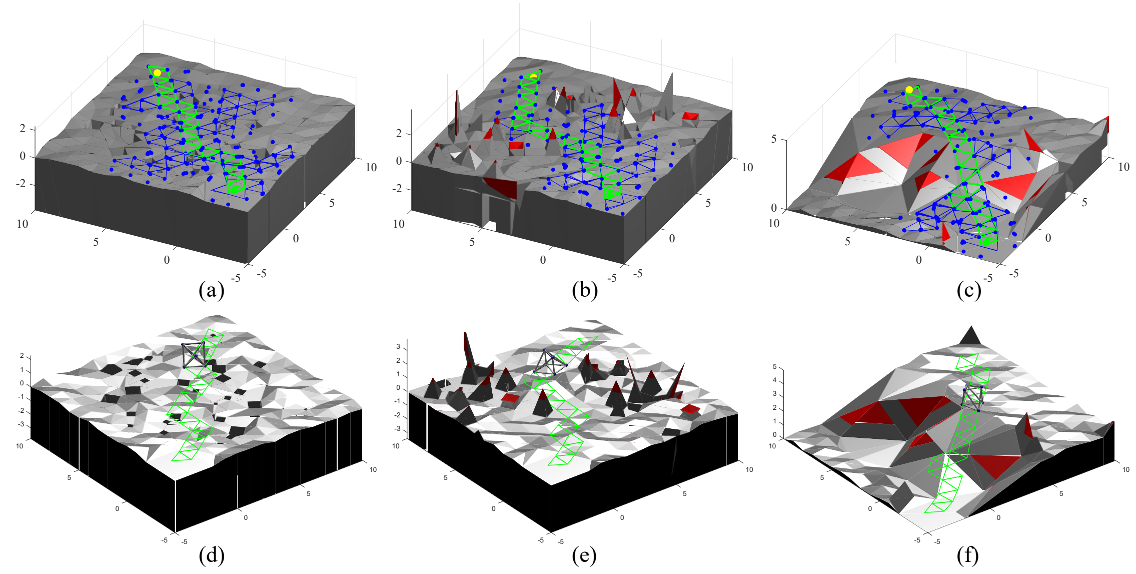

Fig. 7. The simulation results; The support polygon sequences: (a) Environment 1 (Holes); (b) Environment 2 (Spikes); (c) Environment 3 (Hill). The locomotion planning results: (d) Environment 1 (Holes); (e) Environment 2 (Spikes); (f) Environment 3 (Hill).

are prevented by terrain characterization and pre-checking in Algorithm 1, most iterations were able to be finished.

V. Conclusion

In this paper, we present the locomotion planning algorithm for truss robots based on support polygon planning and motion primitive generation. The expanded PRT algorithm can successfully generate support polygons on various terrains. We were able to make the algorithm more efficient by excluding unreachable surfaces with terrain characterization. The motion primitive idea is applied for efficient and reliable planning. The static analysis makes it possible to consider force-related constraints such as friction and internal forces. We proved the performance of the algorithm in various terrains. For future work, we are planning to merge the size-changing planner into 3D locomotion planning [18]. The terrain characterization will be improved to determine unreachable areas and the optimal size of a truss robot to pass through the environment.

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] G. J. Hamlin and A. C. Sanderson, "Tetrobot: A modular approach to parallel robotics," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 42–50, 1997.

[3] 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.

[4] N. S. Usevitch, Z. M. Hammond, M. Schwager, A. M. Okamura, E. W. Hawkes, and S. Follmer, "An untethered isoperimetric soft robot," *Science Robotics*, vol. 5, no. 40, 2020.

[5] A. Spinos, D. Carroll, T. Kientz, and M. Yim, "Topological reconfiguration planning for a variable topology truss," *Journal of Mechanisms and Robotics*, vol. 13, no. 4, p. 040901, 2021.

[6] A. Spinos and M. Yim, "Towards a variable topology truss for shoring," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2017, pp. 244–249.

[7] 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.

[8] 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.

[9] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 634–641.

[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 Robotics and Automation Letters*, vol. 5, no. 2, pp. 813–819, 2020.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[12] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5539–5544.

[13] M. Saboia, V. Thangavelu, W. Gosrich, and N. Napp, "Autonomous adaptive modification of unstructured environments," in *Robotics: Science and Systems XIV*, 2018, pp. 70–78.

[14] 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.

[15] N. S. Usevitch, Z. M. Hammond, and M. Schwager, "Locomotion of linear actuator robots through kinematic planning and nonlinear optimization," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1404–1421, 2020.

[16] F. Collins and M. Yim, "Design of a spherical robot arm with the spiral zipper prismatic joint," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 2137–2143.

[17] S. Park, "Stable rolling locomotion for variable topology truss robot," Ph.D. dissertation, Seoul National University, 2020.

[18] J. Bae, S. Park, M. Yim, and T. Seo, "Polygon-based random tree search algorithm for a size-changing robot," *IEEE Robotics and Automation Letters*, 2021.

[19] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Algorithmic foundation of robotics VII*. Springer, 2008, pp. 507–522.

[20] S. Hirose, "A study of design and control of a quadruped walking vehicle," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 113–133, 1984.