2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Stable, Autonomous, Unknown Terrain Locomotion for Quadrupeds based on Visual Feedback and Mixed-Integer Convex Optimization

Min Sung Ahn, Hosik Chae, Dennis W. Hong[1]

Abstract— **This paper presents a complete motion planning approach for quadruped locomotion across an unknown terrain using a framework based on mixed-integer convex optimization and visual feedback. Vision data is used to find convex polygons in the surrounding environment, which acts as potentially feasible foothold regions. Then, a goal position is initially provided, which the best feasible destination planner uses to solve for an actual feasible goal position based on the extracted polygons. Next, a footstep planner uses the feasible goal position to plan a fixed number of footsteps, which may or may not result in the robot reaching the position. The center of mass (COM) trajectory planner using quadratic programming is extended to solve for a trajectory in 3D space while maintaining convexity, which reduces the computation time, allowing the robot to plan and execute motions online. The suggested method is implemented as a policy rather than a path planner, but its performance as a path planner is also shown. The approach is verified on both simulation and on a physical robot, ALPHRED, walking on various unknown terrains.**

I. INTRODUCTION

As robotics technology further matures and more robots enter environments where they are to co-exist with humans, robots will have to adapt to the various demands of both natural and man-made environments. While wheeled robots provide both stability and efficiency, they become limited when faced with complex and discontinuous terrains. Legged robots provide far better performance in these unknown and potentially challenging terrains.

However, path planning for legged systems is a non-trivial task, especially as the number of legs and the degrees of freedom increases. The increase in the number of legs does provide more stability and is one of the factors behind a surge in multi-legged robots conducting meaningful tasks in man-made environments. Traditionally, these path planning problems have been tackled using a sampling approach, some form of a graph search, or a planner based on optimization.

Sampling based approaches are immensely useful when dealing with problems with high degrees of freedom and constraints like a multi-legged robot [1]. For example, RRTs have been used to reactively path plan in environments with 3D moving obstacles [2], albeit the use of a motion capture system. There are also many cases of A* being used in the space of possible actions and replanning, as unforeseen obstacles are introduced. The downside of these is that they have needed a lot of computation time along with heuristics [3], [4], [5]. Similar and more efficient work has been done using AD* and ARA*, but these also suffer from heuristics playing an important role in their effectiveness [6], [7], [8]. Deits found near optimal solutions for bipeds, which Aceituno-Cabezas extended to multi-legged robots, but a user had to inform the robot of safe regions that it could step on before the optimization could be run [9], [10]. The downside with many of these approaches is that a lot of the times there requires a human operator behind the scenes, meticulously assisting the robot in forms ranging from providing a complete map with heuristic information to carefully designed initial conditions. In these environments without a priori knowledge, if the human is mostly removed from the control loop, even a slightly complex environment will make it difficult for these approaches to be effective.

Fig. 1. ALPHRED walking over an unknown terrain by planning based on sampling and optimization.

In this paper, we utilize a combination of random sampling and a sequence of optimization to systematically plan motions from visual feedback in an unknown environment with only an initial input by a human. Unlike in other approaches, the effort required to design the initial input is very simple–it is simply the final goal position. Afterwards, we rely on visual feedback to provide regions that the robot may be able to use to solve for an optimal footstep sequence and center of mass trajectory. Even if this final

*This research was supported by a grant (code R2016001) from Gyeonggi Technology Development Program funded by Gyeonggi Province.

[1]All authors are with the Department of Mechanical and Aerospace Engineering at the University of California, Los Angeles, CA 90095, USA aminsung@ucla.edu, hosikchae@ucla.edu, dennishong@ucla.edu

goal position is physically impossible, the robot will make its best effort autonomously. By introducing randomness from using a sampling approach in front of a sequence of optimizations, we achieve a probabilistic convergence to the final goal position. However, because of this randomness from sampling, the regions make the entire planner non-optimal when considering the entire path from start to finish, but by taking a policy approach, each footstep and COM motion is optimal given its state.

Contribution: Therefore, our contribution in this paper is primarily in two parts. The first is that we provide a complete method for a multi-legged robot to autonomously path plan given a goal position to head towards, and the problem is treated as a policy problem, where each state's inputs (vision data) result in an optimal next action based on a series of mixed-integer convex optimizations. The second part further utilizes mixed-integer convex optimization to extend the optimal ZMP-constrained COM trajectory originally constrained in 2D into 3D. Optimizations are designed with easy constraints to minimize the solving time to reach a smooth motion in-between steps. In the next section, we give a brief overview on related works on planar extraction and footstep planning.

II. Related Works

Plane detection in robotics have been dealt with in many different ways to provide the robot with valuable information in a timely manner. Geometric features have been extracted from point clouds using 3D Hough transform [11], region growing methods [12], [13], and RANSAC based approaches [14]. Non-conventional approaches such as semi-definite programming approaches [15], methods that use second derivative estimates [16], and stereo-fusion based segmentation have also been shown [17], but they require relatively longer solving times.

There are also a lot of different approaches to general footstep and COM trajectory planning, with some of the approaches already introduced. A decomposition of the planning into global and execution phases have been explored [18], while planners that represent the contact manifold geometrically and decomposing it to reduce the complexity of the planning problem have also been shown [19]. While many of the previously mentioned approaches are constrained to a fixed orientation, non-linear programming based algorithms that also consider orientation in the footstep planning, albeit the lack of optimality, have been proposed [20].

III. TECHNICAL APPROACH

Our approach is conducting plane detection on vision data in a favorable way for the best feasible destination planner and footstep planner, such that the optimal feasible destination and footsteps can be planned autonomously given a goal position. Afterwards, mixed-integer constraints are included into an originally simple quadratic program, to overcome bilinear and nonlinear constraints and solve for a height trajectory while maintaining the ZMP stability criterion.

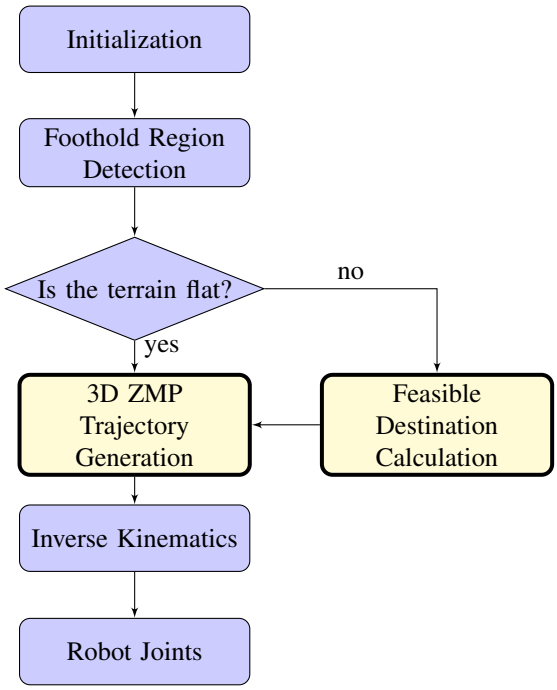


Fig. 2. Different flow depending on flatness of detected terrain. If terrain is determined flat, robot regards all the open areas as safe regions and walks based on the ZMP trajectory. Boxes with thicker border are those that solve optimization problems.

A. Random Sampling of Foothold Regions

When randomly sampling potential foothold regions, there are two criteria that all regions have to satisfy.

1) A randomly sampled region should have dimensions that are greater than a pre-defined minimum.
2) The offset angle between the region's normal vector and the global Z-axis must be less than some maximum value.

To begin with, points in the pixel space with depth value between $depth^{min}$ and $depth^{max}$ are extracted to prevent sampling on points with depth values of no interest. No interest points are points that returned 0, inf, or a depth value over a pre-defined maximum depth value, which could potentially create unwanted polygons that would negatively impact the solving speed of subsequent modules. Then, as long as the number of randomly sampled points and the number of search attempts are below a pre-defined maximum (maximum number of filtered points $n_{p,filt}^{max}$ and maximum number of regions n_{rg}^{max}) a while loop searches through the reduced number of points. An initial point, $p_{rnd,0}$, from the reduced points is randomly chosen, and based on this point, two other points, $p_{rnd,1}$, $p_{rnd,2}$, which are less than η distance from $p_{rnd,0}$, are chosen. A normal vector to these three points is calculated, and to satisfy the second criteria, its offset to the global Z-axis is computed.

If this offset is below the chosen maximum offset, the

vicinity of these three points is considered as a potential region that the robot can safely step on without slipping, since the incline is expected to be a reasonable amount. To guarantee the first criteria which is chosen based on the foot size of the robot, the actual pixel space to search for around the average of the three randomly chosen points, p_{avg}, is calculated. Then, the search dimensions in the pixel space ($w_w,\ w_h$) are calculated and p_{avg} is given to RANSAC [21] as an input to find points that are inliers and make up a polygon. If there are sufficient number of inliers, the algorithm considers the inlier points $p_{inliers}$ as a potential candidate region, and creates a polygon object from these points and appends it to P, a list of polygons. The polygon object then calculates the points making up its convex hull using Graham scan, its polyhedra form of $Ax \leq b$ [22], its normal vector, its center of mass, and its center based on the points of the convex hull. The pseudocode is shown in Alg. 1.

Algorithm 1 Foothold Region Random Sampling

procedure RANDOMFOOTHOLDREGION(p_d, p_{pc})
  $P \leftarrow \{\}$
  $p_d \leftarrow depth^{min} < p_d < depth^{max}$
  while $n_{p,filt} < n_{p,filt}^{max}$ **and** $n_{rg} < n_{rg}^{max}$ **do**
    $n_{inliers} \leftarrow 0$
    $(p_{rnd},\ r) \leftarrow$ `SelectRandomAndEta()`
    if $r < offset_z^{max}$ **then**
      $p_{avg} \leftarrow$ `XYZAverage(`p_{rnd}`)`
      $(w_w,\ w_h) \leftarrow$ `PixelSpaceSearchDim()`
      $(n_{inliers}, \boldsymbol{p}_{poly}) \leftarrow$ `RANSAC(`p_{avg}, w_w, w_h`)`
      if $n_{inliers} > n_{inliers}^{min}$ **then**
        Append $\boldsymbol{p}_{poly}$ to P
        $n_{ftp} \leftarrow n_{ftp} + n_{inliers}$
    $n_{rg} \leftarrow n_{rg} + 1$
  return P

B. Best Feasible Destination based on Approximate Desired Destination

Afterwards, the best feasible destination is solved for based on the goal position provided by the operator and the list of polyhedra P. The goal position can be chosen as an actual final position for the robot to get to, or it can be a point that the robot should continue to plan towards without ever reaching it; i.e. it can also act as a direction that the robot should head towards. Vision data from the previous module will check to ensure that there are feasible foothold regions that satisfy the robot's kinematics at the goal position; if such regions do not exist, the optimization will find the next nearest position for the robot, which becomes the best feasible destination.

1) Objective: To try to satisfy the user's command while also ensuring kinematic reachability and stability, the cost function is designed to have two parts. Its formulation is shown in Eqn. 1 and Eqn. 2:

- A quadratic cost on the distance between the user's desired COM position p_c^d and the best feasible COM position p_c

$$(p_c - p_c^d)^T Q_c (p_c - p_c^d) \tag{1}$$

- A quadratic cost on the distance between the best feasible COM position and the center of the foothold positions f_c

$$(p_c - f_c)^T Q_m (p_c - f_c) \tag{2}$$

2) Constraint-Footstep Assignment to Feasible Regions: The extracted polygons from the previous module are used to find the best feasible destination. This is done by constraining potential footsteps that place the body at the best feasible destination to be assigned a single region, while all regions are required to be assigned one or zero footsteps. The assignment is done by using a binary matrix $\mathscr{H} \in \{0,1\}^{n_{rg} \times n_{legs}}$, where $\mathscr{H}_{r,j} = 1$ when the j-th footstep is assigned to region r. Otherwise, $\mathscr{H}_{r,j} = 0$.

$$\mathscr{H}_{rj} \implies A_r f_j \leq b \tag{3}$$

$$\textstyle\sum_r \mathscr{H}_{rj} = 1 \quad \text{for } j = 1, \cdots, n_{legs} \tag{4}$$

$$\textstyle\sum_j \mathscr{H}_{rj} \leq 1 \quad \text{for } r = 1, \cdots, n_{rg} \tag{5}$$

3) Constraint-Kinematic Reachability: Reachability is taken into account by constraining each foothold to be within R_{nom} from the best feasible destination. Since R_{nom} is the maximum distance that a foot can reach, an absolute value on the difference between the foothold f_j and p_c constrained to be less than or equal to R_{nom} is included:

$$\|f_j - p_c\|_2 \leq R_{nom} \quad \text{for } j = 1, \cdots, n_{legs} \tag{6}$$

C. Footstep Planner to the Best Feasible Destination

Optimal footstep positions are found by using the polygons detected from vision data and the best feasible destination obtained previously. To further lessen the time spent on solving the optimization problem, additional filtering of vision data that would be beneficial to the optimization program is done. Defining v_c as the vector from the current position of the COM to the best feasible destination of the COM, two regions that are parallel and to the left and right of v_c are generated in the form of $Ax \leq b$. This region is created because considering the kinematic reachability constraints of the physical robot, there is little reason to use the complete set of randomly sampled regions that the robot may never be able to reach in the first place. Then, given a set of potential foothold regions, the best feasible destination, and the number of steps to solve for, a mixed-integer convex optimization problem is solved to assign footsteps to regions.

1) Objective: The cost function to be minimized has three main portions that shape how the footsteps should be assigned:

- A quadratic cost on the difference between the center of the feet and the best feasible destination

$$(p_c - f_{c,k})^T Q_g (p_c - f_{c,k}) \tag{7}$$

- A quadratic cost on the relative distance between steps of the same leg

$$(f_{k+n_{legs}} - f_k)^T Q_r (f_{k+n_{legs}} - f_k) \tag{8}$$

- A linear cost on the confidence of the safety of each region

$$c_f P_a H_{a,k} \quad \begin{matrix} \text{for } a = 1, \cdots, n_{rg} \\ \text{for } k = 1, \cdots, n_{fs} \end{matrix} \tag{9}$$

Since mostly the x, y position of the center of the feet is close to the COM position, and since we plan the footsteps before finding an optimal COM trajectory, we put a cost on the distance from the center of the feet configurations to the best feasible destination. We also put a cost on the relative distance between steps such that the robot does not try to take huge steps that may lead to instability. Lastly, the cost on the confidence of the safety of each region, scaled by a confidence factor of c_f, encourages the robot to choose foothold regions that are safer, where safer regions are those with a bigger polygon area P_a, even if it may result in a roundabout path, to ensure a safe locomotion.

2) Constraint-Safe Foothold Regions: A relaxed version of the safe foothold region constraint from the previous section is used (i.e. only Eqn. 3 and 4), as Eqn. 5, which constrains each region to be assigned to at most one footstep, is no longer necessary and multiple footsteps can be assigned to one region.

3) Constraint-XYZ Reachability: Reachability in the XY plane is constrained from the current center of footsteps, where a reachability constraint is enforced per leg by constraining the foot position to be inside a box of dimensions $(bound_x, bound_y)$. This box is represented to have a center at a nominal distance, L_{nom}, away from the footstep's center, and at a nominal angle relative to the x axis of the robot for the respective leg, which is also a function of the heading angle $\theta_{heading}$ as is seen in Fig. 3. The heading angle is simply the angle from the robot to the best feasible destination.

$$\theta_{heading} = Atan2(p_{c,y}, p_{c,x})$$

$$\begin{aligned} l_{nom,x} &= L_{nom} cos(\theta_{heading} + offset_{nom}) \\ l_{nom,y} &= L_{nom} sin(\theta_{heading} + offset_{nom}) \end{aligned}$$

$$\begin{aligned} f_{k,x} &\in f_{c,k,x} + l_{nom,x} \pm \frac{bound_x}{2} \\ f_{k,y} &\in f_{c,k,y} + l_{nom,y} \pm \frac{bound_y}{2} \end{aligned}$$

Reachability in the Z is simply constrained by putting a bound on the maximum offset in the Z direction for the same foot.

$$f_{k,z} \in f_{k-n_{legs},z} \pm bound_z$$

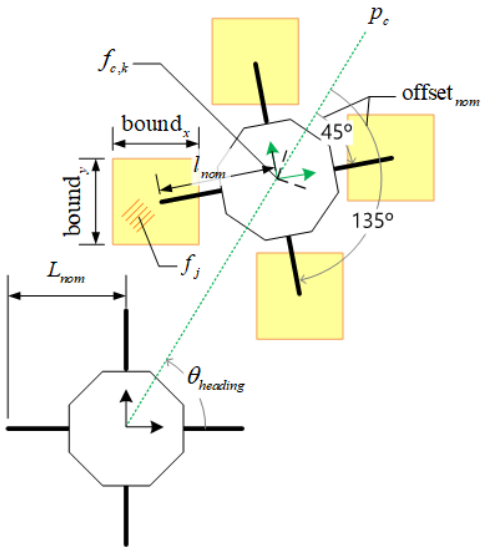


Fig. 3. A graphical representation of the reachability constraint in the XY plane.

D. 3D ZMP Trajectory with Stability

Given the footstep position there are multiple ways to generate a COM trajectory that the robot is to follow. Traditionally, this trajectory is calculated based on a constraint on maintaining the stability of the robot in the sense of the ZMP, as defined in the following definition.

Definition Stability in the sense of the ZMP is when the ZMP, defined as in Eqn. 10, is inside the support polygon created by the support points of the robot.

$$x_{zmp} = x_m - \frac{z_m \ddot{x}_m}{\ddot{z}_m + g} \tag{10}$$

However, to do this, a fixed COM height z_m and zero acceleration in the Z direction $\ddot{z}_m$ is a requirement to prevent bilinear constraints. For variable height terrains such as stairs, a vertical displacement is unavoidable and a simple linear interpolation in the z direction risks violating the stability definition. While non-linear programming solvers can solve such complexities to optimize over the Z trajectory, we approached the problem by approximating the bilinear constraints in a mixed-integer quadratic program, because of its ease of setup, solve time with modern solvers, and a sub-optimal solution being acceptable as long as stability is guaranteed. Given the optimal footstep positions from the previous section, the optimization was formulated to solve for the coefficients of six quintic polynomials in x, y, and z directions that are continuous to the second derivative. The exact formulation of the entire program is shown below.

1) Objective: The chosen objective function was to minimize the cost of the acceleration in all x, y, and z directions for safe locomotion [23].

$$\ddot{x}_1 + \ddot{y}_1 + \ddot{z}_1 + \ddot{x}_2 + \ddot{y}_2 + \ddot{z}_2 \tag{11}$$

2) Constraint-Continuous Trajectory: Because we wanted the robot to follow a smooth, continuous path without jerks in-between steps, a constraint was added such that not only the position, but also the velocity and the acceleration had to be continuous between the two trajectories. Eqn. 12, 13, 14 is also applied for y and z trajectories.

$$x_i(T_i) = x_{i+1}(0) \tag{12}$$

$$\dot{x}_i(T_i) = \dot{x}_{i+1}(0) \tag{13}$$

$$\ddot{x}_i(T_i) = \ddot{x}_{i+1}(0) \tag{14}$$

3) Constraint-ZMP Stability: The ZMP position is calculated by Eqn. 10, and this point is required to be within a support polygon created by three points of the stance feet, with each line of the polygon defined in the form of $px + qy + r = 0$.

4) Approximation of $z_m\ddot{x}_m$: In our case, because the planner is used to also solve for trajectories traversing uneven terrain, it is mandatory for z_m to be a polynomial rather than a constant; consequently, its trajectory is also optimized. However, in such a case, a bilinear constraint is introduced, resulting in having to resolve to using other complex solvers which may take more time to compute. However, this constraint can also be taken care of using approximations such as the McCormick Envelope [24] or by multi-parametric disaggregation technique [25]. In our case, we approximate using a McCormick envelope by defining a new decision variable $w := z_m\ddot{x}_m$, and adding the envelope as the constraints.

$$w_x \geq -z_{m_L}\ddot{x}_{m_L} + z_m\ddot{x}_{m_L} + \ddot{x}_m z_{m_L} \tag{15}$$

$$w_x \geq -z_{m_U}\ddot{x}_{m_U} + z_m\ddot{x}_{m_U} + \ddot{x}_m z_{m_U} \tag{16}$$

$$w_x \leq \ddot{x}_m z_{m_U} - z_{m_U}\ddot{x}_{m_L} + z_m\ddot{x}_{m_L} \tag{17}$$

$$w_x \leq z_m\ddot{x}_{m_U} - z_{m_L}\ddot{x}_{m_U} + \ddot{x}_m z_{m_L} \tag{18}$$

5) Approximation of $\frac{w_x}{\ddot{z}_m+g}$: Now that we have $\frac{w_x}{\ddot{z}_m+g}$, we have a nonlinear constraint, and the McCormick envelope cannot be implemented directly. To overcome this, we define a new variable $k := \ddot{z}_m + g$. Then, a binary matrix $\mathscr{Z} \in \{0,1\}^{N\times S}$ is created, with N as the number of piecewise linear approximations of $\frac{1}{k}$ and S as the number of splines in two consecutive support polygons. Finally, another variable $u = \frac{1}{k}$ is created, and we use $\mathscr{Z}$ to map $\frac{1}{\ddot{z}_m+g}$ to u while preserving convexity. The mapping is done by the following mixed-integer constraints:

$$\mathscr{Z}_{ns} \Rightarrow \begin{cases} k_{m_L} \leq k_{m,s} \leq k_{m_U} \\ u = m_s k_{m,s} + b_s \end{cases}$$

Also, each spline is constrained to a single segment in the linear approximation by Eqn. 19. As an example, in the case of $N = 3$, a segment can be any one of the segments shown in Fig. 4.

$$\sum_{n=1}^{N} \mathscr{Z}_{n,s} = 1 \tag{19}$$

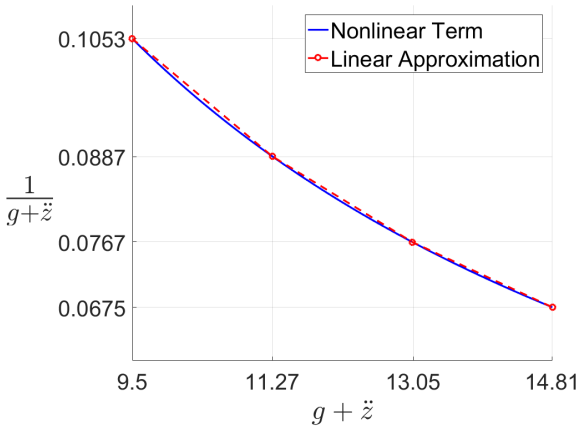


Fig. 4. Example of a piecewise linearization with $N = 3$ for demonstration purposes.

This brings us to the familiar bilinear constraint, where we can define $v := w_x u_x$. The adaptation of the envelope as in Equation 15 $\sim$ 18 can be used. Now, instead of the traditional constraint using the ZMP stability criterion where z_m is to be a constant and $\ddot{z}_m = 0$, z_m can also be optimized for where it does not have to be a constant and $\ddot{z}_m$ does not have to equal zero.

IV. EXPERIMENT

The main goal of this approach was to give more autonomy to the robot by finding a feasible destination from an initial goal position commanded by the operator, and executing a motion plan based on visual feedback. Therefore, we evaluate the robot's performance in terms of its ability to move towards the best feasible destination in an unknown terrain from only a rough goal position commanded by the operator. However, since the majority of this approach is a sequence of processing of data and optimization based on a previous outcome, we also evaluate its performance in terms of computation time. We record the times of the full sequence of scanning for foothold regions to solving for the COM trajectory by summing the runtime of each individual module listed in Section III. The approach was tested on both simulation and verified on a physical robot called ALPHRED (Autonomous Legged Personal Helper Robot with Enhanced Dynamics) [26].

A. Simulation Setup

The entire motion planning sequence was evaluated on three different terrains (flat, stairs, random rough terrain) with the operator giving the same desired goal COM destination of $x = 1.25$m, $y = 1.25$m, and $z = 0.57$m. Note that when this command is given, both the operator and the robot have no terrain knowledge and consequently, no pre-planned future motions. The computer used to run the sequence was equipped with an Intel Core i7-7700HQ (2.80 GHz) and 16 GB of RAM (2400 MHz), running Ubuntu 16.04 and Python 2.7. For all optimization, we used Gurobi [27].

B. Physical Verification

ALPHRED is a modular multi-modal robot with four limbs with each limb having 3 degrees-of-freedom that can be used as both legs and hands depending on the needs of the application. Its end-effector can easily be modified to equip a bar feet and a point feet, and it also has a neck where a camera can be mounted to accumulate vision data. In our testing on both simulation and experiment, point feet was used on the end-effector, and an RGBD camera (Intel Realsense R200) was used. We used the on-board computer on ALPHRED, which is equipped with an Intel Core i5-5250U and 8 GB of RAM.

V. RESULTS

A. Overall Performance

Overall, given the initial goal position by the operator, this process was able to continuously and autonomously find a path towards the best feasible destination by finding feasible regions, selecting the safe ones to plant the feet, and generating a motion plan for the next step. Fig. 6 shows a random sampling of potential foothold regions and Fig. 5 shows the framework executed on ALPHRED for climbing stairs, with the footsteps and COM trajectory shown and executed on both the simulation model and the physical system.

B. Timing Tests

We realize that despite the maturity of state-of-the-art solvers, because we are solving a sequence of mixed-integer programs, it was still important to monitor the time it took for each module to fully finish its task, as the original goal was to produce a smooth motion. Table I shows an average of 1000 samples of running the entire sequence in simulation, with the amount of time it took per module in milliseconds.

TABLE I
TIMING OF MODULES IN MILLISECONDS

Terrain	Vision	GPP	FSP	MP	Total Time
Flat	2.001	5.112	4.533	6.5376	18.1836
Stairs	2.013	4.924	4.236	6.5189	17.6919
Rough	1.843	4.638	3.394	6.4921	16.3671

C. Determination of Best Feasible Destination

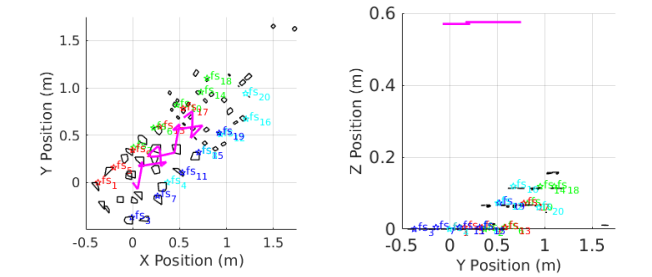
The best feasible destination from feasible regions is shown in Fig. 7 for flat and rough terrain. With flat terrain, the locations of the feasible regions are well aligned, and the best feasible destination is in the direction towards the desired COM position. With rough terrain, the best feasible destination is still found regardless of the irregular distribution of the feasible regions, but it lies on the edges of the support polygon and the modified destination is clearly not in the direction towards the desired destination.

D. Determination of a Confident Path

Given the best feasible destination, a confidence factor c_f in Eqn. 9 determines how conservative the footstep planner should be when planning the footsteps to take. This is more influential when planning long trajectories rather than policies, as the planner could stay conservative for the first few steps and get more aggressive in the later steps, while when the framework is used as a policy, the aggressive behavior may not show up immediately in the best next step. Thus, in the case with trajectories, as can be seen in Fig. 8, one plan shows a roundabout path, but a path with safer regions, while the other plan shows a more direct path, but one with riskier regions. Note that the planner chose to stall at the end with the safe path because it deemed further paths as unsafe.

VI. DISCUSSION

Our results showed that each module functions properly and when integrating the modules together, the robot can



(a) Results seen in XY plane (b) Results seen in XZ plane

(c) Execution in simulation (d) Execution in physical model

Fig. 5. Simulation results for the framework.

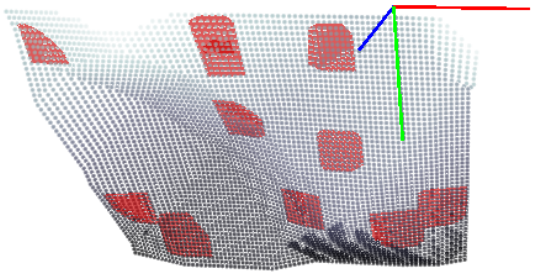
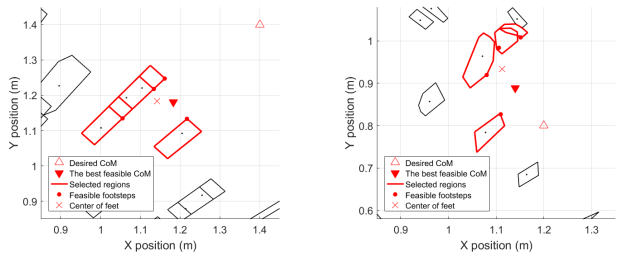


Fig. 6. Detected feasible regions (red polygons) extracted from point cloud data (grayscale points) of a scan of a rough terrain. Three lines colored in red, green and blue represent X, Y, Z axis of the camera, respectively.



(a) Flat Terrain (b) Rough Terrain

Fig. 7. Computed the best feasible destination under (a) flat terrain condition and (b) rough terrain condition. For a given desired COM, the best feasible destination is found by trying to be as close as possible to the desired commanded COM, while considering the robot's reachability and feasible regions.

plan for long steps if needed, but also always optimize for the next best step to take based on randomly sampled regions without delay in-between steps because of quick computation times.

However, in regards to the time it took in each module, it is heavily dependent on the parameters of the vision filter. $n_{p,filt}$ and n_{rg} determine the amount of data that the vision module is to return, and since this data is used by subsequent modules, these parameters affect the computation speed of the entire approach.

Because we can take one step from a single run through the entire sequence, and repeat this process at every step, we could afford to limit the number of polygons extracted from a single snapshot of the point cloud data and ensure fast solving times throughout the entire approach. The timings in Table I are from extracting 10.511 polygons on average over a 1000 snapshots in a terrain with many flat areas, such as stairs. This is the worst case scenario for our chosen $n_{p,filt}$ and n_{rg} in our application, because as the environment has less flat surfaces, less feasible polygons will be returned, and

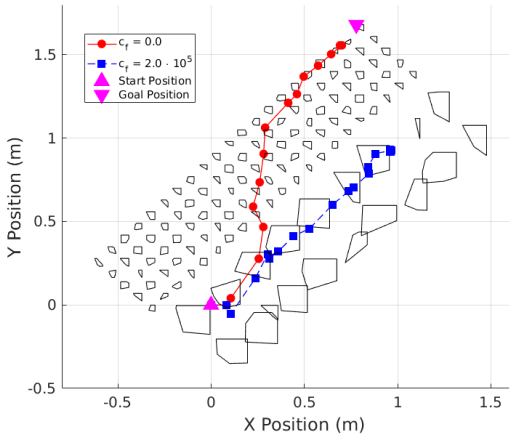


Fig. 8. Different paths depending on the confidence factor c_f. Bigger c_f results in choosing a path with polygons having bigger area, while smaller c_f results in choosing a riskier path that will allow the robot to end closer to the best feasible destination.

since computing the different characteristics of a polygon take up a significant time, overall the amount of time to sample from the vision data once will decrease. However, in situations where the number of polygons decrease, we technically diverge from optimality as probabilistically we may not sample the optimal regions the footstep planner needs to directly move towards the best feasible destination.

In regards to optimizing for the best feasible destination, it played an important role in modifying a command that may be impossible. We wanted the robot to still be able to approach the commanded goal positions to some degree despite a poor command, while also avoid it if the command was impossible. It worked well because even if the human wanted to give a poor command, the planner would make its optimal decision based on what the vision module saw, and provide a modified destination to the footstep planner. Again, as it is based on randomly sampled vision data, despite an optimal path to the user's goal position existing, the destination planner may provide the footstep planner with an incorrect destination, resulting in a sub-optimal path. Additionally, there are no guarantees of stability when the best feasible destination is optimized for. Nonetheless, in subsequent footsteps, new vision information could provide foothold regions where stability may be possible.

The footstep planner also adds a second layer of autonomy and safety. Before the footstep planner, the polygons are considered as feasible regions and that is it. From the footstep planner, the notion of a safe foothold region is introduced with the introduction of a confidence factor. The confidence factor is used to guide the optimization to either choose smaller regions but find a more direct footstep towards the best feasible destination, or take a roundabout path that ensures that the foothold assigned regions' areas are bigger than that of the majority sampled. To test its performance, instead of planning a single set of footsteps to go towards the best feasible destination, we specified an exact number of footsteps to take. In the case when c_f was set to 0, the cost function's confidence terms were eliminated, resulting in a function purely dependent on the distance between the center of the feet sequences and the best feasible destination. Consequently, a straightforward path to the goal was solved for. On the contrary, when c_f was set to 2.0×10^5, the optimization chose bigger regions that would be safer in the sense that the actual footstep could afford to be off from the planned footsteps and the robot would still likely be stable. This would however, come at the cost of the final position not reaching the best feasible destination.

Furthermore, the COM trajectory solver is able to solve for 3D trajectories, while still maintaining ZMP stability. However, depending on how many splines are to be created and how much linear approximation of $\frac{1}{\ddot{z}_m+g}$ we plan on doing, better trajectories may be achieved, but at the price of longer solve times. In our situation, because we were commanding the robot to find the next best action in an unknown terrain, and because of the conservative approaches taken in the previous modules, we expected $\ddot{z}_m + g$ to not deviate too much from 9.81. Therefore, we could limit the

number of piecewise segments in the linear approximation and still solve for stable COM trajectories in 3D.

VII. CONCLUSION & FUTURE WORK

This work introduces a framework for multi-legged robots to autonomously plan and travel over unknown terrains, given an initial goal position. A randomly sampled vision data is passed to a series of mixed-integer programs where the robot can autonomously optimize for modified goal positions and footsteps, and solve a COM trajectory in all x, y, and z directions, while keeping stability as a necessary constraint and safety as a cost. The framework can be used as a policy, where vision data is processed and the next best step is taken based on the current state of the robot, to achieve a reactive planner robust to intermediate errors. However, it can also be used as a path planner that solves for multiple sets of footsteps.

The framework is able to successfully navigate unknown terrains with only the initial goal position commanded, as the vision module and optimizations modify and adapt the goal position and footsteps such that they are feasible and safe. The downside is that the approach is highly dependent on the performance of the sensors and that an optimal path is never guaranteed. Nonetheless, constraints in stability and the ability to realize an approach of sampling and optimizing after every step because of fast computation times allow the approach to eventually navigate to the best feasible destination.

In the future, we plan on further exploring the idea of "confidence" in our optimization, where more than just information on potential foothold regions' areas are included, such as the material of the region or the dynamic state of the entire robot. What is considered to be a "reasonable terrain" for a robot with certain kinematics and dynamics are also to be explored. Furthermore, the framework can still be enhanced, as currently, the approach constrains the orientation of the body, which makes the workspace limited. With this extension, an even more sophisticated path planner over unknown terrains could be achieved, where the robot chooses footsteps that it could not previously find with a fixed orientation.

REFERENCES

[1] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[2] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida, "Real-time replanning using 3D environment for humanoid robot," *IEEE-RAS International Conference on Humanoid Robots*, pp. 584–589, 2011.

[3] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning," *Proceedings of the 2007 7th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS 2007*, pp. 196–202, 2008.

[4] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade, "Vision-guided humanoid footstep planning for dynamic environments," *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, vol. 2005, pp. 13–18, 2005.

[5] P. Karkowski, S. Oßwald, and M. Bennewitz, "Real-time footstep planning in 3D environments," *IEEE-RAS International Conference on Humanoid Robots*, pp. 69–74, 2016.

[6] A. Hornung and M. Bennewitz, "Adaptive level-of-detail planning for efficient humanoid navigation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 997–1002, 2012.

[7] D. Maier, C. Lutz, and M. Bennewitz, "Integrated perception, mapping, and footstep planning for humanoid navigation among 3D obstacles," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2658–2664, 2013.

[8] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "An Optimization Approach to Rough Terrain Locomotion An Optimization Approach to Rough Terrain Locomotion," pp. 3589–3595, 2010.

[9] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Febru, pp. 279–286, 2015.

[10] B. Aceituno-Cabezas, J. Cappelletto, J. C. Grieco, and G. Fernandez-Lopez, "A Generalized Mixed-Integer Convex Program for Multilegged Footstep Planning on Uneven Terrain," 2016. [Online]. Available: http://arxiv.org/abs/1612.02109

[11] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani, "Recognising Structure in Laser Scanner Point Clouds," *Information Sciences*, vol. 46, no. April 2016, pp. 1–6, 2004.

[12] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 3378–3383, 2008.

[13] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, p. 5284, 2010.

[14] J. Biswas and M. Veloso, "Fast Sampling Plane Filtering, Polygon Construction and Merging from Depth Images," *RGB-D Workshop at RSS 2011*, pp. 2–7, 2011.

[15] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," *Springer Tracts in Advanced Robotics*, vol. 107, pp. 109–124, 2015.

[16] R. Hulik, V. Beran, M. Spanel, P. Krsek, and P. Smrz, "Fast and accurate plane segmentation in depth maps for indoor scenes," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1665–1670, 2012.

[17] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-December, pp. 881–888, 2015.

[18] K. Hauser, T. Bretl, J. C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[19] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *The International Journal of Robotics Research*, no. X, pp. 1–11, 2016.

[20] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. Darpino, R. Deits, M. Dicicco, D. Fourie, and et al., "An architecture for online affordance-based perception and whole-body planning," 2014.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[23] P. B. Wieber, "Viability and predictive control for safe locomotion," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 1103–1108, 2008.

[24] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems," *Mathematical Programming*, vol. 10, no. 1, pp. 147–175, 1976.

[25] J. P. Teles, P. M. Castro, and H. A. Matos, "Multi-parametric disaggregation technique for global optimization of polynomial programming problems," *Journal of Global Optimization*, vol. 55, no. 2, pp. 227–251, 2013.

[26] J. Hooks and D. Hong, "Implementation of a versatile 3d zmp trajectory optimization algorithm on a multi-modal legged robotic platform," *International Conference on Intelligent Robots*, 2018.

[27] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com