Pose Prediction for Mobile Ground Robots in Uneven Terrain Based on Difference of Heightmaps

Stefan Fabian, Stefan Kohlbrecher, and Oskar von Stryk

***Abstract*— For traversing uneven terrain in degraded environments, determining the static stability and consequently the tip-over risk of a mobile ground rescue robot is fundamental for planning and evaluation of paths. This paper presents a novel iterative geometric method that reduces the problem of robot pose prediction to two-dimensional image-processing operations by introducing the concept of a robot heightmap. The presented method requires only geometrical and mass information extracted from the widely used unified robot description format (URDF) to compute the robot heightmap, which makes it transferable to a wide range of mobile robot platforms without modification. We demonstrate that the approach accurately predicts the real robot's 6D pose at the input x-y-coordinates. Runtimes allowing the evaluation of poses in the order of ten thousand poses per second show that the method is computationally efficient enough to be used in online path planning.**

I. Introduction

Catastrophic events such as the recent chemical explosion in Beirut, Lebanon, August 2020, can lead to hazardous environments where exposure poses a high risk to human rescue forces. Deploying rescue robots can help mitigate the risk by allowing remote exploration and manipulation reducing the time human rescue forces are exposed to these high-risk environments. However, the deployment in such environments is highly challenging as robot hardware and software need to account for the challenging conditions, e.g. unstructured, uneven terrain with high tip-over risk and limited radio connectivity.

In these environments, robot-terrain interaction becomes increasingly relevant. Algorithms that take the terrain structure as well as the robot configuration into account are essential for the deployment of (semi-)autonomous rescue robots to become viable.

One particularly active field of research where the robot-terrain interaction is receiving increased attention is the field of UGV path planning for robots deployed in rescue scenarios or extra-planetary missions. Generally, in mobile robot path planning, the objective is to find a traversable path from a start position to a goal position. In rescue scenarios, this search is subject to a soft real-time requirement due to the exploratory nature of rescue missions where the terrain is previously unknown and mission time is critical. A high number of potential locations and orientations on a discretized or sampling-based graph have to be evaluated to plan an (approximately) optimal path. Hence, for online planning, the evaluation of a single location has to be computationally cheap on mobile robot computing hardware. As computer hardware advances, the algorithms have moved on from efficient methods such as planning on occupancy grids[4], which greatly simplify the problem by using a binary classification into free and blocked cells in a fixed-resolution 2D grid, to using handcrafted statistics as a heuristic to estimate whether a surface is traversable or not [3][7], or using statistics to label sections into predefined classes such as *stairs*, *floor*, *wall*, *surmountable obstacle* etc. [6][8]. Both the traversability estimates and the predefined classes can not capture scenarios where a robot may traverse an obstacle in one configuration or orientation but not in another and represent a generally conservative limit in the actions the robot may take.

Brunner et al. divide the models for robot-terrain interaction into four major approaches [2]: machine learning, full dynamic physics simulation, quasi-static physics simulation, and iterative geometric methods (IGM). Of these four, only the last two do not require a training phase and suitable terrain features that can represent a variety of terrain structures (machine learning) nor accurate mechanical models of the robot and the terrain (full dynamic physics simulation). Brunner et al. propose an iterative geometric method to efficiently compute the pose and support polygon – which is defined as the convex hull of the robot's supporting ground contacts. The proposed algorithm fits a plane to the local ground, computes the lowest point in each relevant geometry – e.g. each flipper or wheel – as contact candidates, and finds the configuration of three contact candidates where the projection of the center of mass (COM) is closest to the triangle formed by the contact candidates with the condition that the other contact candidates must be above the plane formed by the three selected candidates. The proposed approach was evaluated against a quasi-static physics simulation using the **O**pen **D**ynamics **E**ngine (ODE). It was demonstrated to be capable of predicting around twenty-thousand robot poses per second compared to ten poses per second with ODE showing that usage in online path planners is possible.

Efficient and accurate algorithms for the estimation of robot-terrain interaction are crucial for the employment in online path planning but the application is not limited to path planning. Other relevant areas of research include but are not limited to whole-body control – where the trajectory for all joints along a given input path, e.g., across an obstacle, is optimized with regard to the robot's stability and

Stefan Fabian and Oskar von Stryk are with the Technical University of Darmstadt, Computer Science Department, Simulation, Systems Optimization and Robotics Group, Germany.
{fabian, stryk}@sim.tu-darmstadt.de
Stefan Kohlbrecher is with Energy Robotics GmbH, Germany.
kohlbrecher@energy-robotics.com

ground contact points – and operator assistance functions, e.g. to visualize the currently estimated static stability or the predicted static stability along the current path extrapolated from the current operator inputs.

In [10], an optimization-based IGM is used to predict the robot pose and ground contacts in the context of whole-body control. Since whole-body control optimizes both the ground contact using reconfigurable tracks if available and the location of the COM if possible, e.g. using a heavy multi-dof arm, the performance and accuracy greatly depend on the method employed to predict the robot's pose and ground contacts.

In this paper, we introduce a novel iterative geometric approach to efficiently compute the robot pose and contact points on a terrain heightmap using only information extracted from the robot's **U**nified **R**obot **D**escription **F**ormat (URDF) making it applicable to a wide range of mobile robots without modification. The presented approach reduces the dimensionality of the problem to two-dimensional image-processing operations that can be performed efficiently on modern consumer-grade robot hardware. In contrast to [2], the approach computes both the support polygon and all contact points enabling the usage for traction estimates and makes no assumptions about the terrain structure. To obtain ground truth contact points and heightmaps from the Gazebo simulator for evaluation purposes, two plugins were developed and are available as open-source contributions on GitHub.

The paper is structured as follows: In Section II, the concept of a robot heightmap is introduced and the algorithms we have developed to efficiently compute the robot heightmap for a robot described using a URDF file are presented. Section III details how this robot heightmap can be used to compute the robot's pose and ground contacts. Subsequently, in Section IV, the presented method is evaluated on the real robot depicted in Fig. 1 on a set of ramps of varying inclination with different orientations and while crossing a set of ramps on a $22\,\mathrm{cm}$ step.

II. Robot Heightmap

Heightmaps are a compact two-dimensional representation of the local terrain where each discretized grid cell contains a single value representing the maximum elevation above a ground plane. A particularly attractive property, in the context of this work, is that single cell and submap access can be achieved in $\mathcal{O}(1)$.

To efficiently predict the robot pose on a terrain heightmap, we introduce the concept of a two-dimensional heightmap of the robot as $\mathrm{HM}(\mathcal{N},\mathcal{N}) \rightarrow \mathcal{R}$ to reduce the dimensionality of the problem. Each cell in the robot heightmap represents the minimal distance from a plane parallel to the x-y-plane, e.g. the bottom of the robot's axis-aligned bounding box, to a part of the robot. Fig. 1 visualizes the heightmap for the robot platform used in the evaluation of the pose prediction approach proposed in this paper as a point cloud. Subtracting the terrain heightmap from the robot heightmap yields the contact map where each cell represents the distance from the ground to the robot. From the contact map, we extract the contact points and the support polygon as the convex hull of the contact points. For an unstable pose, the rotation axis can be determined by using a stability measure to identify the unstable axis of the support polygon and the necessary rotation can be estimated from the contact map.

The heightmap of the robot is generated by iterating through the components of the robot and for each component computing the distance to the bottom of the robot's axis-aligned bounding box and if that distance is smaller than the currently stored value, the value is updated with the new minimal distance.

A. Coordinate Transform

While the robot is given in the URDF coordinate system, the heightmap spans in the x-y-plane from $[0,0,0]^T$ to $[\lceil \mathrm{BB}_x/\rho \rceil, \lceil \mathrm{BB}_y/\rho \rceil, 0]^T$ where $\mathrm{BB}_{x/y}$ are the dimensions of the bounding box BB in x- and y-direction and ρ is the resolution - i.e. the quadratic cell width - of the heightmap. Since the heightmap represents the bottom of the bounding box, the robot has to be transformed such that the center of the bounding box's bottom is moved to the center of the heightmap. Since the rotation of the robot is done before the computation of the bounding box, this simplifies to a translation t_{HM}

$$t_{\mathrm{HM}} = c_{\mathrm{HM}} - c_{\mathrm{BB}} = \begin{bmatrix} \lceil \mathrm{BB}_x/\rho \rceil \cdot \rho/2 \\ \lceil \mathrm{BB}_y/\rho \rceil \cdot \rho/2 \\ 0 \end{bmatrix} - c_{\mathrm{BB}}$$

where c_{HM} is the center of the heightmap and c_{BB} is the center of the bounding box.

B. Heightmap Generation

The heightmap is generated based on the three primitives that are commonly used in the URDF's collision model: spheres, boxes, and cylinders. Extending the algorithm to meshes is straight-forward but due to the significantly higher computational cost, it is not recommended to use meshes in the collision model.

1) Sphere: Due to the rotation-invariance, the sphere is the most simple form of the three primitives. Given a sphere with center $\vec{c}$ and a radius r, the orthogonal projection to the x-y plane of the heightmap is a circle with radius r. This can be used to efficiently compute the values for each cell

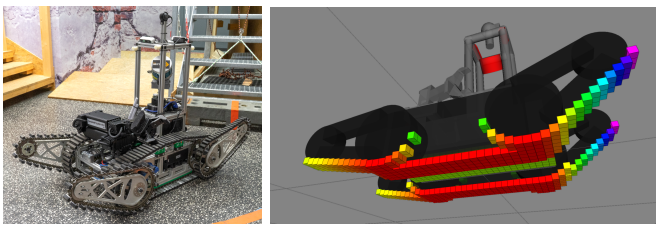

Fig. 1. Left: The robot platform used in this work. A tracked robot with two chassis tracks and four reconfigurable flippers. Right: The URDF model and the generated heightmap with a resolution of $5\,\mathrm{cm}$ converted to a point cloud for visualization.

in the heightmap. We iterate y in discrete steps from $y_{\min}$ to $y_{\max}$ which are computed as

$$y_{\min} = \left\lfloor \frac{(\vec{c}_y - r)}{\rho} \right\rfloor \qquad y_{\max} = \left\lceil \frac{(\vec{c}_y + r)}{\rho} \right\rceil$$

Using the circle equation $r^2 = x^2 + y^2$, we can compute $x_{\min}$ and $x_{\max}$ using the effective radius in x-direction r_x for each $y_i \in [y_{\min}, y_{\max}]$.

$$y_r = \begin{cases} y_i \cdot \rho + \frac{\rho}{2} & , y_i \cdot \rho \leq \vec{c}_y - \frac{\rho}{2} \\ y_i \cdot \rho - \frac{\rho}{2} & , y_i \cdot \rho \geq \vec{c}_y + \frac{\rho}{2} \\ \vec{c}_y & , \text{otherwise} \end{cases}$$

$$r_x = \sqrt{r^2 - (y_r - \vec{c}_y)^2}$$

$$x_{\min} = \left\lfloor \frac{(\vec{c}_x - r_x)}{\rho} \right\rfloor \qquad x_{\max} = \left\lceil \frac{(\vec{c}_x + r_x)}{\rho} \right\rceil$$

Here, y_r is snapped to the cell border closest to the center of the sphere or the center itself in the discretized grid because that is where the distance to the x-y plane is the smallest. In the next step, the same is done for each $x_k \in [x_{\min}, x_{\max}]$ and the minimal distance d_z is computed using the sphere equation $r^2 = x^2 + y^2 + z^2$ as

$$x_r = \begin{cases} x_k \cdot \rho + \frac{\rho}{2} & , x_k \cdot \rho \leq \vec{c}_x - \frac{\rho}{2} \\ x_k \cdot \rho - \frac{\rho}{2} & , x_k \cdot \rho \geq \vec{c}_x + \frac{\rho}{2} \\ \vec{c}_x & , \text{otherwise} \end{cases}$$

$$d_z = \vec{c}_z - \sqrt{r^2 - (y_r - \vec{c}_y)^2 - (x_r - \vec{c}_x)^2}$$

2) Box: The computation of the heightmap for a box requires processing one to three discrete sides depending on the orientation.

Based on

$$N = |C_b| = |\{\vec{c} : \vec{c}_z \leq z_{\min} + \delta; \vec{c} \in C\}|$$

where C is the set of corners of the box and N is the number of corners with z equal to the minimum value of z among all corners plus a small delta δ for numerical stability, we split the computation into three cases of increasing complexity: $N = 4$, $N = 2$, and $N = 1$.

For $N = 4$, there is only one constant height difference across the entire box which was calculated before as $z_{\min}$. Thus, the calculation of the heightmap values simplifies to efficiently iterating an arbitrarily rotated (around the z-axis) rectangle.

In our case, the y-axis is iterated[1] and we compute the minimum and maximum value for x using a scanline approach. We store the corners of the rectangle (counter-) clockwise as $[a, b, c, d]$ and determine the corner $i_{\min}$ with the smallest y-value resolving conflicts with the smaller x-value. Then we obtain the highest corner as $i_{\max} = (i_{\min}+2) \mod 4$, and the left and right corner by computing which one is left of the line connecting the lowest and highest corner.

[1]Which one is more efficient depends on the memory storage order.

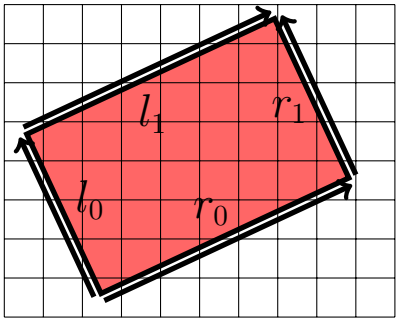

Fig. 2. An example projection for $N = 4$ and the contour lines.

The problem of computing the minimum and maximum value of x for each y now simplifies to following two sets of two lines. From lowest via left to highest and from lowest via right to highest. Based on the bottom $\vec{p}_b$, top $\vec{p}_t$, left $\vec{p}_l$, and right $\vec{p}_r$ corner we compute the line gradients

$$l_0 = \frac{\vec{p}_{l,x} - \vec{p}_{b,x}}{\vec{p}_{l,y} - \vec{p}_{b,y}} \qquad r_0 = \frac{\vec{p}_{r,x} - \vec{p}_{b,x}}{\vec{p}_{r,y} - \vec{p}_{b,y}}$$
$$l_1 = \frac{\vec{p}_{t,x} - \vec{p}_{l,x}}{\vec{p}_{t,y} - \vec{p}_{l,y}} \qquad r_1 = \frac{\vec{p}_{t,x} - \vec{p}_{r,x}}{\vec{p}_{t,y} - \vec{p}_{l,y}}$$

Using these gradients, we can compute the value of the rectangle borders for each y switching between r_0 / l_0 and r_1 / l_1 when we pass the relevant corner. As illustrated in Fig. 2, the row centers may not coincide with the minimum x-value for that row. Hence, a correction term is added

$$o_{l,0} = -\frac{|l_0|}{2} \quad o_{l,1} = -\frac{|l_1|}{2} \quad o_{r,0} = \frac{|l_0|}{2} \quad o_{r,1} = \frac{|l_1|}{2}$$

To use this simple form of offset, the start of the lines has to be adjusted such that they start at the center of a row. Special handling has to be employed for the start and end of each line.

For $N = 2$, we have two sides facing downwards that need to be identified. We propose indexing the corners as in Fig. 3. Using this structure, three indices of the form $\{n, n{+}1, n{+}2\}$ always form a side of the rectangle. The same holds for three indices of the form $\{n, n{+}2, n{+}4\}$. The two missing corners to constrain the relevant sides are split into two cases. If the two bottom corners have increasing indices of the first form ($d = 1$), the two missing corners are given as the highest index ± 2. If the indices are of the second form ($d = 2$), they are given as the highest index $+2$, and -1 if the highest index is odd and $+1$, otherwise.

Analogous to the algorithm in $N = 4$, we can traverse the rectangle using the rectangle representation

$$\vec{p}_r = \vec{p}_a + \alpha \cdot \vec{u} + \beta \cdot \vec{v} \qquad 0 \leq \alpha, \beta \leq 1$$

$$\vec{u} = \vec{p}_b - \vec{p}_a \qquad \vec{v} = \vec{p}_d - \vec{p}_a$$

to compute the value for each cell of the robot heightmap

$$\text{HM}\,(x_k, y_i) = \vec{p}_{a,z} + \alpha_{i,k} \cdot \vec{u}_z + \beta_{i,k} \cdot \vec{v}_z$$

where $\alpha_{i,k}$ and $\beta_{i,k}$ are a linear interpolation from $\alpha_{i,\min}$ to $\alpha_{i,\max}$ and $\beta_{i,\min}$ to $\beta_{i,\max}$ obtained by solving

$$\vec{c}_{i,\cdot,x} = \vec{p}_{a,i,x} + \hat{\alpha}_{i,\cdot} \cdot \vec{u}_x + \hat{\beta}_{i,\cdot} \cdot \vec{v}_x$$
$$\vec{c}_{i,\cdot,y} = \vec{p}_{a,i,y} + \hat{\alpha}_{i,\cdot} \cdot \vec{u}_y + \hat{\beta}_{i,\cdot} \cdot \vec{v}_y$$

for $\vec{c}_{i,\min}$ and $\vec{c}_{i,\max}$ which are the cells with minimal and maximal x for a given y.

To obtain $\alpha_{i,\min}$, $\alpha_{i,\max}$, $\beta_{i,\min}$ and $\beta_{i,\max}$, an offset has to be added to $\hat{\alpha}$ and $\hat{\beta}$ in order to represent not the center but the minimum of the cell. The offset is obtained as a constant independent of x or y using

$$\begin{aligned} &\min_{\alpha_o,\beta_o} \alpha_o \cdot \vec{u}_z + \beta_o \cdot \vec{v}_z \\ \text{s.t.} \quad &|\alpha_o \cdot \vec{u}_x + \beta_o \cdot \vec{v}_x| \leq 0.5 \\ \text{and} \quad &|\alpha_o \cdot \vec{u}_y + \beta_o \cdot \vec{v}_y| \leq 0.5 \\ \alpha_{i,\cdot} = \hat{\alpha}_{i,\cdot} + \alpha_o \quad &\beta i,\cdot = \hat{\beta} i,\cdot + \beta \end{aligned}$$

Since the objective function is linear and the constraint space is rectangular, at least one of the corners is part of the solution space. At the start or end of a row, the offset may be outside of the rectangle. In that case, the solution is one of the intersections of the rectangle with the cell border or the corner of the rectangle.

For $N = 1$, we have three sides facing downwards. Using the structure of the indices, if the index m of the lowest corner is odd, the sides are given by the combinations $\{m-2, m, m+2\}$, $\{m-2, m, m-1\}$, and $\{m-1, m, m+2\}$. If the index of the lowest corner is even, the combinations are $\{m-2, m, m+2\}$, $\{m+1, m, m+2\}$, and $\{m-2, m, m+1\}$. Iteration of the rectangles and determination of the minimal z-value for each cell is done analogously to $N = 2$.

3) Cylinder: A cylinder is given by its radius on the x-y-plane and the length or height in z-direction.

The points on the surface of a cylinder with the center at the origin are given by

$$\begin{aligned} r = \sqrt{x^2 + y^2} \quad &, \text{if} \quad -\frac{l}{2} \leq z \leq \frac{l}{2} \\ z = \pm\frac{l}{2} \quad &, \text{if} \quad \sqrt{x^2 + y^2} \leq r \end{aligned}$$

where r is the radius and l the length in z-direction.

Depending on the cylinder's orientation, the projection is either a circle of constant height, a rectangle of varying height, or a combination of two ellipses and a rectangle. The case can be determined using the projected length in z-direction of the unit z-vector.

If the projected length is 1, the projection is a circle with constant z-value. The computation simplifies to the iteration of a circle analogously to the algorithm described for the sphere shape in II-B.1 without the need to compute a location-dependent z-value.

If the projected length is 0, the projection is a rectangle with the z-value dependent on the distance to the projected central axis. The length of the rectangle is given by the length of the cylinder and the width by twice the radius. The rectangle is given by the projection of the bottom and top center points $\vec{c}_b = [0, 0, -l/2]$ and $\vec{c}_t = [0, 0, l/2]$. Using $\vec{v} = \vec{c}_t - \vec{c}_b$ we can obtain the other line of the rectangle $\vec{u}$ as the one perpendicular to $\vec{v}$ with the constraint that $\vec{u}_z = 0$. The corners can then be obtained as:

$$\vec{a} = \vec{c}_b - r \cdot \vec{u} \qquad \vec{b} = \vec{c}_b + r \cdot \vec{u}$$
$$\vec{c} = \vec{c}_t + r \cdot \vec{u} \qquad \vec{d} = \vec{c}_t - r \cdot \vec{u}$$

The iteration can be done analogously to the rectangle iteration for the box in II-B.2. It is extended by a line from $\vec{c}_b$ to $\vec{c}_t$ which is used to determine the minimal distance $d_{\min}$ of a cell to this central line. The heightmap value can then be obtained as:

$$d_z = \vec{c}_z - \sqrt{r^2 - d_{\min}^2}$$

For the third case, a projected length between 0 and 1 results in two ellipses connected by a rectangle with height values computed analogously to the rectangle projection case with an additional z-gradient along the central axis.

Extending the formula for angled cylinders (see Fig. 4) gives

$$h(x, y) = \frac{\sqrt{r^2 - d(x_k, y_i)^2}}{\sin\gamma}$$

$$\text{HM}(x_k, y_i) = \vec{c}_{b,z} + \frac{\vec{v}_{x,y} \cdot \left(\begin{bmatrix} x_k \\ y_i \end{bmatrix} - \vec{c}_b \right)}{|\vec{v}_{x,y}|_2^2} \cdot \vec{v}_z - h(x_k, y_i)$$

This equation contains both a linear and a non-linear term, hence, computing the minimum is highly expensive and we incur a small error by only approximating the minimal distance by using a fixed position in the cell constrained to be inside the cylinders bounds.

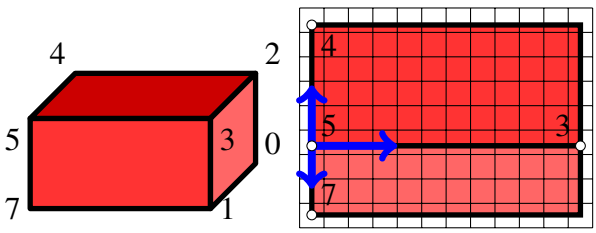

Fig. 3. Left: Corner indices. Right: Projection for $N = 2$.

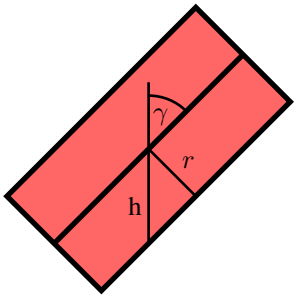

Fig. 4. Cylinder with an angle of γ degrees to the z-axis and radius r.

For the top ellipse, the heightmap values are computed with the same equations as the rectangle and we only need to iterate the ellipse row-wise. Using the ellipse equation $Ax^2 + Bxy_i + Cy_i^2 = 1$ with the center at the origin, we obtain $x_{\min}$ and $x_{\max}$ for each y_i as

$$A = \left(\frac{\cos^2\alpha}{a^2} + \frac{\sin^2\alpha}{b^2}\right) \quad C = \left(\frac{\sin^2\alpha}{a^2} + \frac{\cos^2\alpha}{b^2}\right)$$

$$B = 2\cos\alpha\sin\alpha\left(\frac{1}{a^2} - \frac{1}{b^2}\right)$$

$$x = -\frac{B}{2A}y_i \pm \sqrt{\left(\frac{B}{2A}\right)^2 y_i^2 - \frac{C}{A}y_i^2 + \frac{1}{A}}$$

where a and b are computed using that the projection of the central axis $\vec{v}$ is the minor axis and the perpendicular vector $\vec{u}$ is the major axis with a length of r. The length of the minor axis is given by the cosine of the angle γ between the untransformed and the transformed z-unit vector $\hat{e}_z$. The angle α is obtained as the angle between the x-axis and the major axis $\vec{u}$.

$$\gamma = \cos^{-1}\hat{e}_z \quad a = r \quad b = \cos\gamma \cdot r \quad \alpha = \cos^{-1}\frac{\vec{u}_x}{|\vec{u}|}$$

The $y_{\min}$ and $y_{\max}$ for the ellipse are obtained as the extrema of a different formulation of the ellipse equation with the center of the ellipse at (c_x, c_y)

$$x = c_x + a\cdot\cos\alpha\cdot\cos t - b\cdot\sin\alpha\cdot\sin t$$
$$y = c_y + a\cdot\sin\alpha\cdot\cos t + b\cdot\cos\alpha\cdot\sin t$$
$$t_x = \tan^{-1}\left(-\frac{b}{a}\cdot\tan\alpha\right) + n\cdot\pi$$
$$t_y = \tan^{-1}\left(\frac{b}{a}\cdot\cot\alpha\right) + n\cdot\pi$$

The bottom ellipse is excluded from the rectangle and instead, the value for each point in the bottom ellipse is computed using that the position projected onto the vector $\vec{w}$ – which is obtained by rotating $\vec{v}$ by 90 degrees around $\vec{u}$ – is the height value:

$$\text{HM}(x_k, y_i) = \frac{\left(\begin{bmatrix}x_k\\y_i\end{bmatrix} - \vec{c}_{b/t,x,y}\right)\cdot\vec{w}_{x,y}}{|\vec{w}_{x,y}|_2^2}\cdot\vec{w}_z + \vec{c}_{b/t,z}$$

where $\vec{c}_{b/t}$ is either $\vec{c}_b$ or $\vec{c}_t$ depending on which has a smaller z-value.

III. Pose Prediction

Using the previously generated robot heightmap, the pose is predicted in two alternating steps: First, the robot heightmap is computed, and the contact points and the resulting support polygon are estimated, then, the stability for each axis of the support polygon is calculated. If one of those is unstable, in the second step, the rotation around that axis is predicted and we go back to the first step. This process is repeated until all axes of the support polygon are stable.

1) Contact Map: Contact points are estimated using the contact map M_{contact} which is obtained by subtracting the terrain heightmap map representing the robot's surroundings from the robot heightmap. The contact points are the cells which have a value smaller or equal to the minimum of the contact map plus a contact delta which represents a trade-off between the accuracy of the contact point estimation and robustness against errors in the terrain heightmap due to sensor noise present in the sensor data used to compute the map.

$$\mu_{\text{contact}} = \min M_{\text{contact}}$$

$$\{p : (x,y) \in p, M_{\text{contact}}(x,y) \leq \mu_{\text{contact}} + \delta_{\text{contact}}\}$$

If there is only one contact point, the rotation axis is orthogonal to the line from the contact to the projected COM. If there are two, they form the rotation axis. Otherwise, using Andrew's monotone chain algorithm [1], the convex hull of the contact points is computed to obtain the support polygon. The static stability can be estimated with any stability measure capable of binary stability classification. In this work, the force-angle stability measure (FASM) by Papadopoulos et al. [11] is used which computes the angle between the robot's COM and each axis of the support polygon. If the angle is negative, the robot is unstable and would tip over that axis, otherwise, the algorithm terminates. The process of computing the contact map and subsequently the contact points, support polygon, and tip-over axis is illustrated in Fig. 5

2) Rotation Estimation: If the currently predicted pose is not stable, we estimate how far the robot would tip over the unstable axis until it makes ground contact.

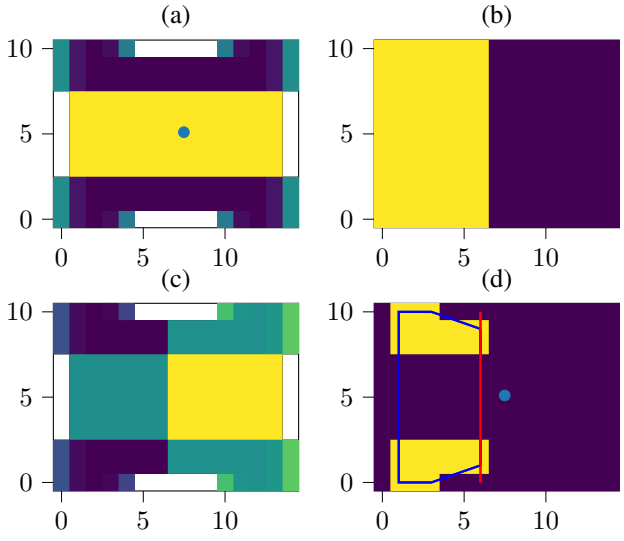

Fig. 5. (a) The robot heightmap with the projection of the COM. Colored from small cell values in dark blue to high values in yellow. White cells have no value. (b) The ground heightmap at a step. (c) The contact map obtained by subtracting the ground from the robot. (d) The contact points in yellow, the support polygon in blue, and the axis of rotation in red.

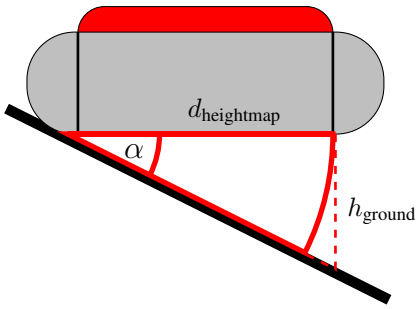

Fig. 6. Rotation on a ramp with slope α.

The rotation angle is obtained by building a triangle between the rotation axis and each cell in the heightmap. We obtain the angle as the minimum over all cells

$$\tilde{\alpha} = \text{atan2}\left(h_{ground},\ d_{heightmap}\right)$$

Please note that this simplification has a small systematic error as indicated by the dashed line in Fig. 6. Since the rotation axis in effectively all cases does not pass through the origin, the origin of the robot has to be moved as well to clear the rotation-induced translation.

IV. Results

The accuracy of the introduced method for predicting the robot pose was evaluated on the real robot in the scenarios depicted in Fig. 7 and the stability predictions were evaluated in a simulated scenario depicted in Fig. 9.

a) Real Robot Scenarios: The ramp scenario was repeated for three inclinations at 5°, 12°, and 20°. For each inclination, the robot was rotated 360° in steps of 22.5° and the pose was recorded. In the elevated ramp scenario, the robot was driven across the elevated ramps manually while using the reconfigurable flippers to test how the approach handles complex poses where the robot is only supported by its flippers. The robot was stopped several times along the path and the pose and joint states were recorded to minimize the influence of dynamic effects on the pose.

Ground truth poses were obtained using an HTC Vive tracking system. As stated in [9], the sample-to-sample RMS of the tracking system is in the sub-millimeter and 0.01° range but there is a systematic offset in both position and orientation due to a slanted reference plane that changes after loss of tracking. However, the pointcloud data used to create the map was also recorded in the localization frame canceling out possible errors due to systematic offset and experiments were constrained to a small area which should minimize the risk of tracking loss.

The heightmap required for the pose prediction was generated using the pointcloud from a rotating Velodyne VLP-16 and the open-source elevation_mapping software [5]. It should also be noted that the presented approach predicts the pose in which the robot would fall if dropped at a given location without non-linear dynamic effects. Since we do not know the input positions that would fall to the recorded positions, we instead do not update the origin after the rotation step and instead keep the x-y position fixed at the cost of possibly requiring more iterations to converge or not converging at all.

For each of the 16 orientations in each ramp scenario and 28 poses that were collected while traversing the elevated ramps, the input pose was generated as the x-y-coordinates of the recorded pose and the extracted rotation around the z-axis. Fig. 8 shows the distribution of the angular error – here defined as the minimal rotation required to get from the predicted orientation to the ground truth orientation – for each scenario in three different resolutions using a contact threshold of 0.5 cm. Because the ramp is a flat surface and the robot ground contact surfaces can be approximated as a plane, different resolutions are not expected to have a big influence on the accuracy of the pose prediction in this scenario. The difference in angular error for the ramp scenarios at different resolutions can largely be explained by map inaccuracies.

In Table I, the mean and maximum angular error is given for different contact thresholds. Pose predictions that did not converge to a stable support polygon within 10 iterations were stopped prematurely. This occurred for less than 4% of the 225 input poses at a contact threshold of 0.25 cm and decreased steadily to 0% at 2 cm. As mentioned before, this is caused by keeping the x-y-coordinates fixed which can result in the prediction algorithm rotating forth and back between two unstable configurations due to the implicit translation when ignoring the rotation induced translation.

Smaller thresholds reduce the error in the rotation angle estimation because h_{ground} is computed from the minimum value but the rotation axis may be up to the contact threshold above the minimum value. However, if the threshold is chosen too small, the error in the estimation of the rotation axis increases as the contact points become more susceptible

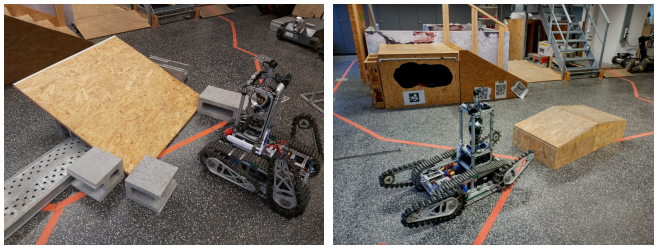

Fig. 7. Evaluation scenarios. Left: A ramp with an inclination of 20°. Right: Elevated ramps with a step height of 22 cm.

TABLE I

Pose prediction angular error and convergence rate.

Contact threshold	Mean	Std. Dev.	Max
0.25 cm	1.45°	0.96°	8.17°
0.50 cm	1.34°	0.65°	3.64°
0.75 cm	1.49°	0.86°	6.23°
1.00 cm	1.46°	0.78°	3.89°
1.50 cm	1.83°	1.05°	9.37°
2.00 cm	2.12°	1.23°	9.37°

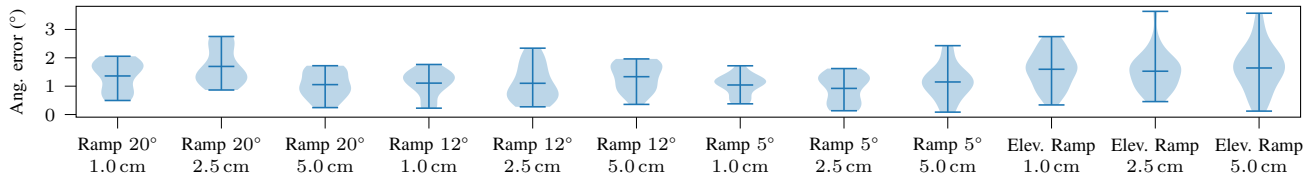

Fig. 8. The angular error plotted for each scenario and the specified heightmap resolutions. The top and bottom bar mark the minimal and maximal angular error. The middle bar represents the mean error and the width represents the density of the error distribution at that value.

to noise in the terrain height values.

Another point to consider is that the $\pm 3\,\mathrm{cm}$ range error of the VLP-16 specified in the sensor's datasheet does not provide precise enough data for the map error to be within the small threshold that is necessary for an accurate rotation angle estimate. Hence, we found that it is generally preferable to use a small threshold for rotation estimation and a higher threshold as a stopping criterion and to compute the final support polygon and stability. With this adaptation, after the first iteration, the support polygon is computed for the higher threshold and only if it is not stable, the support polygon for the smaller rotation contact threshold is computed and the rotation is estimated. This results in a more accurate support polygon and faster computation at the cost of a slightly higher angular error. For example, using a contact threshold of $0.5\,\mathrm{cm}$ for the first iteration and rotation estimation and a threshold of $2.0\,\mathrm{cm}$ for the stopping criterion, the mean error is at $1.54° \pm 1.08°$.

b) Simulated Scenario: As capabilities to obtain ground truth contact points to test the stability estimates of the presented approach were not available, it was evaluated with the physics simulation engine **O**pen **D**ynamics **E**ngine (ODE) used in the robotics simulator Gazebo 9 as a reference. The simulation scenario depicted in Fig. 9 features a step, rough terrain made of boxes of varying height, and a ramp. The path of the robot straight across the set of obstacles was simulated and the poses, contact points, and a ground truth heightmap of the scenario were obtained using the open-source package *hector_ground_truth_gazebo_plugins* [2] developed in the context of this work.

Fig. 10 shows the predicted and simulated roll and pitch, and the stability calculated using the FASM for a map resolution of $5\,\mathrm{cm}$. While the angular error is low with a mean of $1.93°$ and a std. dev. of $\pm$ $2.33°$ (angular error on flat grounds excluded) except for the rough terrain which is the worst case due to the existence of multiple valid poses with low stability, the stability plot is of greater interest as it expresses whether the stability of the robot can be accurately assessed.

The stability graph contains an additional third line with the estimated stability. This is due to the contact points obtained from the simulation being limited to points that are exactly in contact with a surface which leads to the robot often having only three contact points despite being on flat ground which leads to jumps in the stability. Hence, an estimate of the stability was added which uses the presented method to compute only the contact points for the simulated pose. The extremum of the estimated stability at step 98 in Fig. 10 is caused by the simulated robot being off the ramp on flat ground but the contact estimation still overlapping with the ramp. To make the stability values comparable, they are normalized to the stability value on flat ground.

The prediction ran single-threaded on a 6th generation Intel i7 6700K (released in 2015) running at $4.2\,\mathrm{GHz}$. The prediction of the entire path with a length of $5\,\mathrm{m}$ took between $5\,\mathrm{ms}$ at a resolution and step length of $5\,\mathrm{cm}$ and $50\,\mathrm{ms}$ at a resolution and step length of $1\,\mathrm{cm}$. At a rate of 10.000 to 20.000 poses per second depending on the heightmap resolution and the complexity of the terrain, the approach is sufficiently efficient to be used in local online path planning. Detailed timings of the individual predictions are given in Table II.

The timings are in the same order of magnitude as the IGM method presented in [2] where the authors also measured the average time to estimate a single pose for an iRobot Packbot and a Telerob Telemax using ODE at $100\,\mathrm{ms}$–$300\,\mathrm{ms}$. While the timings are not directly comparable as they were not obtained on the same machine, the difference can be expected to be less than one order of magnitude. The robot used for the evaluation is comparable in the complexity of the

[2] https://github.com/tu-darmstadt-ros-pkg/hector_ground_truth_gazebo_plugins

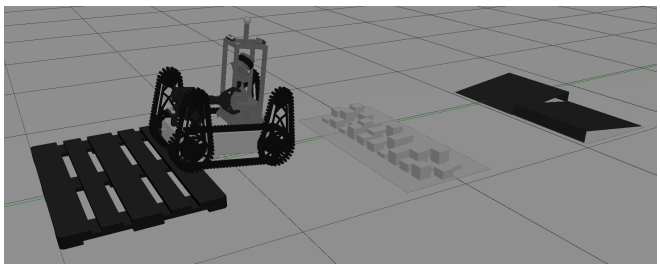

Fig. 9. Simulation scenario consisting of a pallet, rough terrain, and a ramp.

TABLE II

TIMING RESULTS OF HEIGHTMAP POSE PREDICTION.

Resolution	Positions	Mean	Std. Dev.	Maximum
1.0 cm	520	81.33 µs	±86.13 µs	537.24 µs
2.5 cm	208	45.06 µs	±44.69 µs	268.23 µs
5.0 cm	104	47.73 µs	±40.36 µs	157.18 µs

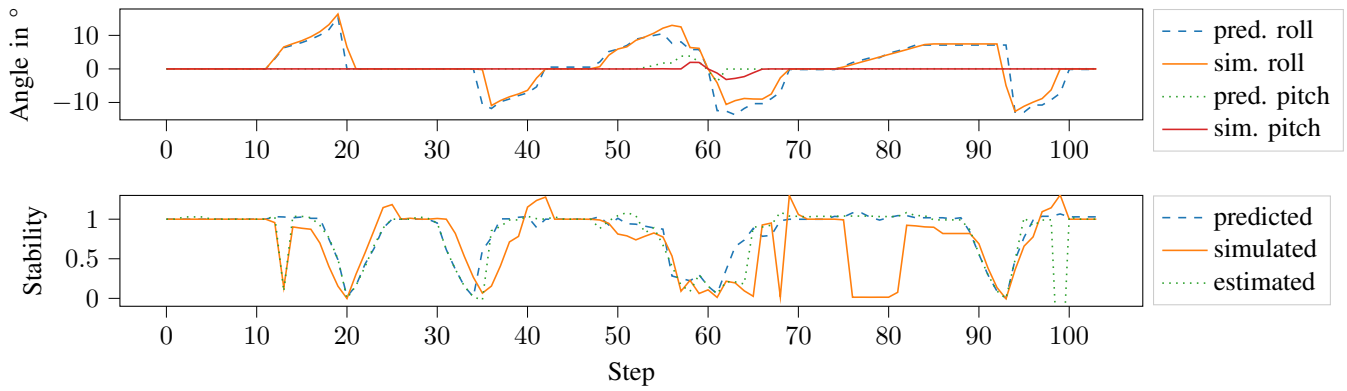

Fig. 10. The top graph plots the predicted roll (dashed blue) and pitch (dotted green) against the simulated roll (solid orange) and pitch (solid red) for each 5 cm increment along the simulated path (Step). The bottom graph plots the predicted stability (dashed blue) against the stability calculated from the simulated contact points (solid orange) and estimated contact points for the simulated pose (dotted green).

geometric model to the Telemax and the processing was not further optimized by limiting the heightmap computation to the components that are most likely to have ground contact. However, the computed heightmaps were cached to allow reusing heightmaps for the same (discretized) orientations.

V. Conclusions

In this paper, the concept of a robot heightmap and algorithms to efficiently compute the robot heightmap from the robot's URDF have been introduced. Based on the concept of a robot heightmap, a novel iterative geometric method for the prediction of the robot pose on a heightmap has been introduced. From the difference between a robot heightmap and the terrain heightmap, the pose and ground contact points are computed iteratively. Our experiments show that the presented method significantly outperforms a common simulation-based approach in terms of computation time while providing for local online path planning sufficiently accurate estimations of the robot's static stability. While similar in terms of performance to the IGM presented by [2], our approach does not assume the local terrain to be flat and provides the means for a traction estimate as it returns all discretized contact areas. The usage of image-processing operations allows further performance optimizations by making use of graphics accelerators and the usage of the defacto standard robot description format URDF enables the support for a wide range of robot platforms using tracked or wheeled locomotion without adaptation. Furthermore, we provide the Gazebo plugin package *hector_ground_truth_gazebo_plugins* used to obtain ground truth data in the simulation as open source. In future work, we intend to apply the developed method in online path planning and both active and feedback operator assistance functions where the approach may be used to predict the pose and stability of the robot at future positions extrapolated from the current control inputs and give an early warning if the predicted stability falls below a safety threshold.

VI. Acknowledgement

This work has been co-funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY centre. Research presented in this paper has been supported in parts by the German Federal Ministry of Education and Research (BMBF) within the subproject "Autonomous Assistance Functions for Ground Robots" of the collaborative A-DRZ project (grant no. 13N14861).

References

[1] A.M. Andrew. "Another efficient algorithm for convex hulls in two dimensions". In: *Information Processing Letters* 9.5 (1979), pp. 216–219.

[2] Michael Brunner et al. "Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots". In: *Robotics and Autonomous Systems* 63 (2015), pp. 89–107. DOI: https://doi.org/10.1016/j.robot.2014.09.003.

[3] F. Colas et al. "3D path planning and execution for search and rescue ground robots". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2013, pp. 722–727.

[4] A. Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989), pp. 46–57.

[5] Péter Fankhauser, Michael Bloesch, and Marco Hutter. "Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization". In: *IEEE Robotics and Automation Letters (RA-L)* 3.4 (2018), pp. 3019–3026. DOI: 10.1109/LRA.2018.2849506.

[6] F. Ferri et al. "Point cloud segmentation and 3D path planning for tracked vehicles in cluttered and dynamic environments". In: *Proc. of the 3rd IROS Workshop on Robots in Clutter: Perception and Interaction in Clutter*. 2014.

[7] Tobias Klamt and Sven Behnke. "Anytime hybrid driving-stepping locomotion planning". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 4444–4451.

[8] M. Menna et al. "Real-time autonomous 3D navigation for tracked vehicles in rescue environments". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 696–702.

[9] D. C. Niehorster, L. Li, and M. Lappe. "The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research". In: *i-Perception* 8.3 (2017).

[10] M. Oehler, S. Kohlbrecher, and O. von Stryk. "Optimization-based planning for autonomous traversal of obstacles with mobile robots". In: *International Journal of Mechanics and Control (JoMaC)* (2020), pp. 33–40.

[11] EG Papadopoulos and Daniel A Rey. "A new measure of tipover stability margin for mobile manipulators". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 4. 1996, pp. 3111–3116.