Steppability-informed Quadrupedal Contact Planning through Deep Visual Search Heuristics

Max Asselmeier[1], Patricio A. Vela[1] and Ye Zhao[1]

***Abstract*— In this work, we introduce a method for predicting environment *steppability* – the ability for a legged robot platform to place a foothold at a particular location in the local environment – in the image space. This novel environment representation captures this critical geometric property of the local terrain while allowing us to exploit the computational benefits of sensing and planning in the image space. We adapt a primitive shapes-based synthetic data generation scheme to create geometrically rich and diverse simulation scenes and extract ground truth semantic information in order to train a steppability model. We then integrate this steppability model into an existing interleaved graph search and trajectory optimization-based footstep planner to demonstrate how this steppability paradigm can inform footstep planning in complex, unknown environments. We analyze the steppability model performance to demonstrate its validity, and we deploy the perception-informed footstep planner both in offline and online settings to experimentally verify planning performance.**

I. Introduction

In 2004 and 2005, the Defense Advanced Research Projects Agency (DARPA) launched the Learning Applied to Ground Vehicles (LAGR) [1] program and the Learning Locomotion (L2) program, respectively [2]. The primary goals of the LAGR program were to make advancements in machine perception including robust object detection and nearsighted sensing for wheeled platforms [3]–[6] whereas the L2 program was concerned with sophisticating motor control and motion planning techniques [7]–[9] for legged systems and subsequently eschewed perception and mapping through the provision of high-accuracy terrain meshes.

From the commencement of the LAGR program to the current day, the concept of terrain traversability has been heavily explored through the lens of wheeled and treaded platforms. High performance image space-based detection and planning methods have been developed that enable real world deployment [10]–[12]. However, with regards to legged platforms, the state of the art in planning and navigation [13]–[15] has largely leveraged environment representations involving Cartesian frame global or robot-centric 2.5D occupancy and height maps, contrasting the perception space-based methods commonly seen for wheeled platforms. While such representations simplify downstream decision-making, they are computationally burdensome for a naviga-

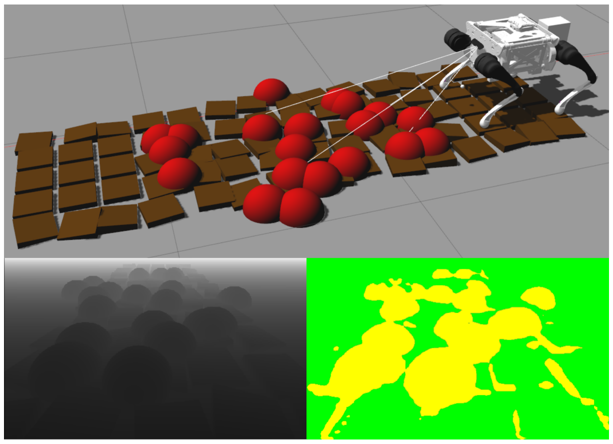


Fig. 1: Unitree Go1 performing autonomous footstep planning in a stepping stones environment. Below, the depth image and steppability mask of the scene are pictured. Within the mask, green corresponds to steppable regions and yellow corresponds to passable regions.

tion framework at the local level due to the need to perform frequent point cloud processing steps such as projection, transformation, and plane estimation. Due to these steps, these representations require GPU acceleration to run online.

In recent years, research efforts have explored how to evaluate legged traversability through the image space [16], but to the authors' awareness, no efforts have been made towards evaluating *steppability* — which captures the geometric properties of the local environment that allow for stable footholds — in the image space. Furthermore, while perception-informed search heuristics have proven to greatly improve graph search timing and performance for manipulation tasks [17]–[19], they have received less attention for the task of legged locomotion. Therefore, in this work, our motivations are twofold: (a) explore how this notion of image space steppability can be predicted through a learned network, and (b) integrate this steppability network into a navigation framework to perform perception-informed footstep planning. In summary, our main contributions are:

1) Introduction of a novel image space-based legged environment representation for steppability
2) Adaptation of a primitive shapes-based synthetic data generation technique to the task of learning steppability
3) Integration of the learned steppability model into an interleaved search and optimization foothold planner
4) Experimental validation of the learned steppability representation and the perception-informed foothold planner in simulation

[1]M. Asselmeier, P.A. Vela, and Y. Zhao are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30308, USA. `mass@gatech.edu`

*The work of Max Asselmeier is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039655. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

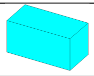
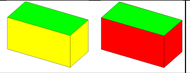
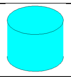
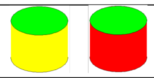
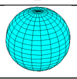
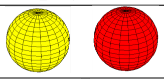
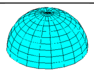
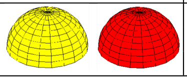
II. Methodology

A. Dataset Generation

First, we will detail how we generate the simulation scenes that we use to extract the synthetic steppability data. Here, we adapt a primitive shapes-based technique that was previously used for manipulation tasks [20], [21]. In this work, the authors hypothesize that the geometry of graspable objects can be decomposed into a set of primitive shapes where each primitive shape class has a particular family of effective grasps. The ground truth labels of these objects can then be ascertained through the color of the primitives within the simulation scene. We perform a similar process here, but instead utilize class labels concerning steppability. The synthetic scenes used for training data are assembled according to a key set of design parameters that we detail now. Example scene data including depth images, ground truth labels, and model predictions is shown in Figure 3.

1) Primitive Shape Classes: We use four primitive shape classes to build scenes: *Cuboid*, *Cylinder*, *Sphere*, and *Semisphere*. The Cuboid class is parameterized by length l, width w, and height h, the Cylinder class is parameterized by x-dimensional radius r_x, y-dimensional radius r_y, and height h, and lastly, the *Sphere* and *Semisphere* classes are parameterized by radius r. Visualizations of each shape class along with design parameter ranges are shown in Table I.

TABLE I: Primitive Shape Classes and Visualizations

Primitive Shape Class	**Geometry Visualizations**	**Label Visualizations**	**Parameter Range (unit: m)**
Cuboid			$l \in [0.2, 1.0]$ $w \in [0.1, 0.50]$ $h \in [0.05, 0.25]$
Cylinder			$r_x \in [0.10, 0.50]$ $r_y \in [0.10, 0.50]$ $h \in [0.05, 0.25]$
Sphere			$r \in [0.025, 0.05]$
Semisphere			$r \in [0.025, 0.05]$

2) Primitive Shape Steppability Policies: Each primitive shape class is assigned a mesh-based steppability policy that defines which faces of the shapes should be assigned which labels. The three labels used are:

- **Steppable (Green):** can support a stable foothold
- **Passable (Yellow):** can not support a stable foothold, but can be stepped over by a foot swing trajectory
- **Non-passable (Red):** can not support a stable foothold and can not be stepped over by a foot swing trajectory

For Cuboids and Cylinders, the top face is labeled as steppable due to its flat horizontal geometry. If the height of the primitive exceeds a maximum swing height for the robot leg, in this work defined as $h_{\max} = 0.10$ m, then all vertical faces are labeled as non-passable. Otherwise, the vertical faces are labeled as passable. For Spheres and Semispheres, the entire primitive is labeled as non-passable if the diameter and radius respectively exceed $h_{\max}$. Otherwise, the entire primitive is labeled as passable.

3) Primitive Shape Pose: The six-dimensional pose of each primitive shape can be set according to desired scene attributes. For instance, scenes can be set to feature clusters of primitive shapes localized within a particular region of the scene, the primitives can be scattered all throughout the scene, or the poses can be overwritten to accept manually defined entries if the user wants to create more contrived scenes that include structures such as staircases or stepping stones. The z-dimension of all shapes is restricted so that all shapes are placed on support surfaces, and the orientations of cuboids and cylinders are restricted to ensure that the face labeled as steppable remains as the top face in the scene.

4) Camera pose: The six-dimensional pose of the camera placed within each simulation scene can also be parameterized to emulate expected real-world circumstances for onboard sensing. Given that our desired application is quadrupedal locomotion, we set the camera at a height of $z = 0.325$ m and pitch it downwards by 30° to approximate the pose of the depth camera attached to our quadruped. To capture multiple frames of a single scene, the camera is set to follow a prescribed trajectory that approximates how a quadruped's torso would move through a real world environment. Gaussian noise is also applied to all six pose dimensions to represent the jitter that the camera would experience during deployment.

5) Scene Environment: Lastly, the overall synthetic environment that the primitive shapes are placed within can also be controlled. In this work, we randomly select between indoor environments that include walls and a ceiling and outdoor environments that include an infinite horizon.

B. Model Training

For model training, we use the off-the-shelf DeepLabV3+ [22] model from the Detectron2 deep learning library [23]. To construct the dataset, 500 scenes were generated with 5 frames each, making for a total of $2,500$ images. In total, dataset generation took roughly 10 hours. The generated data was put into 80%/10%/10% splits of 2,000, 250, and 250 images for training, validation, and test subsets respectively. Model training was performed on an Intel Xeon W-2223 CPU at 3.60GHz along with an NVIDIA T1000 GPU, with a trial taking 37 minutes. Plots of training loss and intersection-over-union can be seen in Figure 2.

C. Steppability-informed Contact Planning

Now, we detail how the proposed steppability model can be used to inform footstep planning. We provide a brief overview here, and more details can be found at [24].

We incorporate this steppability model into our existing interleaved graph search and trajectory optimization-based footstep planner. To plan a discrete sequence of footholds, we perform a search over a mode transition graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ [25]. Each vertex $v \in \mathcal{V}$ represents a partial stance in which a proper subset of the quadruped's feet are in contact with a unique combination of steppable objects in the environment.

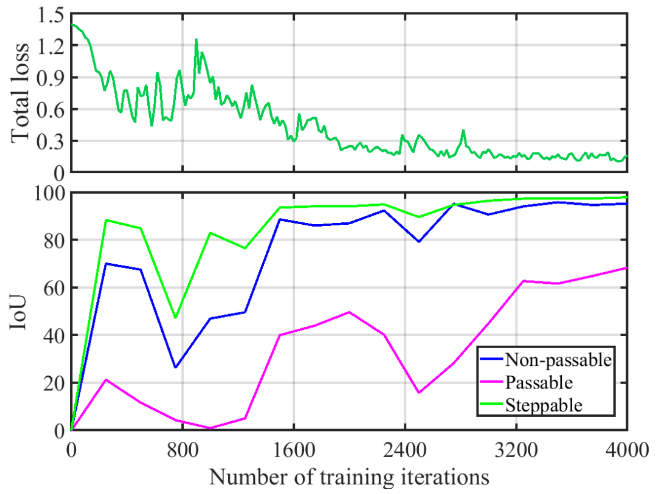


Fig. 2: Results of training. The top figure shows the total training loss over the training iterations. The bottom figure shows the intersection-over-union (IoU) of the three semantic classes over the training iterations.

Each vertex v represents a mode family Ξ. Within each mode family, there is an infinite set of modes $\xi \in \Xi$ where each mode ξ represents a particular set of positions along the steppable objects where contact is made. A set of continuously varying coparameters $\boldsymbol{\chi}$ parameterize these contact positions.

Each edge $e \in \mathcal{E}$ represents a transition between two partial stances which itself is a full stance in which all feet are in contact. For a transition between source mode $\xi_i = \langle \Xi_i, \chi_i \rangle$ and destination mode $\xi_{i+1} = \langle \Xi_{i+1}, \chi_{i+1} \rangle$, the graph edge $e = (\xi_i, \xi_{i+1})$ is assigned the weight

$$\begin{aligned}\Delta c(\xi_i, \xi_{i+1}) = w_{\mathcal{D}} \cdot \mathcal{D}^{\Xi_i, \Xi_{i+1}}(\boldsymbol{\chi}_i, \boldsymbol{\chi}_{i+1}) + \\ w_d \cdot d_{\text{CoM}}(\xi_i, \xi_{i+1}) + w_\tau \cdot d_\tau(\xi_i, \xi_{i+1}) + \\ w_{\text{step}} \cdot d_{\text{step}}(\xi_i, \xi_{i+1}),\end{aligned} \tag{1}$$

where the distribution $\mathcal{D}^{\Xi_i, \Xi_{i+1}}(\boldsymbol{\chi}_i, \boldsymbol{\chi}_{i+1})$ captures the kinodynamically-aware cost of transitioning from ξ_i to ξ_{i+1}, $d_{\text{CoM}}(\xi_i, \xi_{i+1})$ is the Euclidean distance between nominal CoM positions for ξ_i and ξ_{i+1}, $d_\tau(\xi_i, \xi_{i+1})$ is the deviation of nominal CoM positions for ξ_i and ξ_{i+1} from a guiding torso path, and $d_{\text{step}}(\xi_i, \xi_{i+1})$ is a proposed steppability-informed weight. All terms are weighted by positive scalars to assign relative importance.

This graph search returns a discrete sequence of footholds that are then passed to a whole body trajectory optimization (TO) program to generate a full trajectory. The optimal cost values obtained from this TO program are then used to update the experience-based distribution $\mathcal{D}^{\Xi_i, \Xi_{i+1}}(\boldsymbol{\chi}_i, \boldsymbol{\chi}_{i+1})$ which is obtained offline.

D. Steppability heuristic

The mode transition graph is constructed under the assumption that we have access to the poses of the objects in the environment that we want to plan footholds on, but we assume that the poses of obstacles are unknown. We then rely on this proposed perception-informed steppability term for reactive obstacle avoidance.

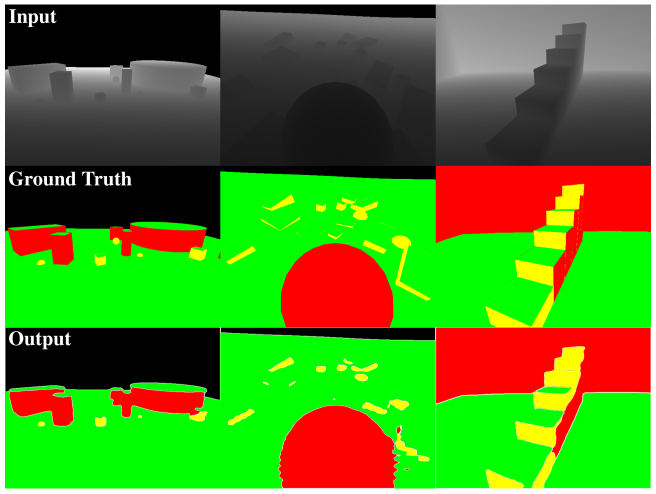


Fig. 3: Example outputs of learned steppability model. The top row depicts examples of input depth images to the model. The middle row depicts the ground truth steppability labels of the corresponding column's input depth image. The bottom row depicts the model outputs for the corresponding column's input depth image.

Prior to triggering the mode transition graph search, the learned model is queried to obtain the current steppability mask $\mathcal{I}_{\text{step}}$. This mask contains steppability labels for the current view of the environment. Then, during the graph search, when a new edge is visited, the stance foothold positions of the edge's transition can be projected into the steppability mask to ascertain information regarding the quality of the candidate foothold positions.

From the graph edge e, we extract the world frame positions $\mathbf{p}_c^l \in \mathbb{R}^3 \quad \forall l = 1, 2, 3, 4$ of the center of each of the four transition footholds. To determine the correct pixel to query for its steppability label, we then perform pinhole camera projection

$$\begin{bmatrix} x_c^l \\ y_c^l \\ w \end{bmatrix} = \mathbf{K} \cdot \begin{bmatrix} \mathbf{R}_{CW}^t & \mathbf{t}_{CW}^t \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_c^l \\ 1 \end{bmatrix}, \tag{2}$$

where $\begin{bmatrix} x_c^l & y_c^l & w \end{bmatrix}^T$ are homogeneous image coordinates, $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the intrinsic camera calibration matrix, and $\mathbf{R}_{CW}^t \in \mathbb{R}^{3\times3}, \mathbf{t}_{CW}^t \in \mathbb{R}^3$ are the rotation and translation from the world frame to the camera frame at time t. Finally, the homogeneous image coordinates of the candidate footholds are obtained through the simple conversion

$$\mathbf{x}_c^l := \begin{bmatrix} x_c^l / w \\ y_c^l / w \end{bmatrix}. \tag{3}$$

Based on the steppability label returned for the foothold pixel $\mathbf{x}_c^l$, we assign a weight of

$$d_{\text{step}}^l(\xi_i, \xi_{i+1}) = \begin{cases} 1000, & \text{if } \mathcal{I}_{\text{step}}(\mathbf{x}_c^l) \rightarrow \text{non-passable} \\ 100, & \text{if } \mathcal{I}_{\text{step}}(\mathbf{x}_c^l) \rightarrow \text{passable} \\ 5, & \text{if } \mathbf{x}_c^l \text{ not in frame} \\ 1, & \text{if } \mathcal{I}_{\text{step}}(\mathbf{x}_c^l) \rightarrow \text{steppable} \end{cases}. \tag{4}$$

Lastly, the total steppability weighting term is taken as the sum across all footholds

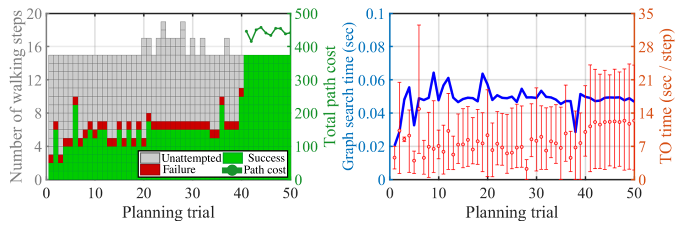
$$d_{\text{step}}(\xi_i, \xi_{i+1}) = \sum_l d^l_{\text{step}}(\xi_i, \xi_{i+1}). \quad (5)$$

To account for the finite size of the quadruped's feet, we perform a step similar to that in [16] where we additionally check the steppability labels of world frame positions that are offset from $\mathbf{p}^l_c$ in the $x-$ and $y-$direction by the radius of the foot and add these to the total steppability weight.

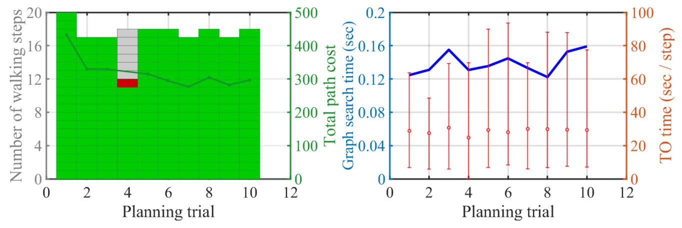
III. Experimental Results

A. Offline Experience Accumulation

First, we demonstrate the added steppability term significantly expedites the offline experience accumulation process. For this experiment, the quadruped is deployed in the environment shown in Figures 1 and 5 where the robot must maneuver across several spherical obstacles scattered along the surfaces of the various stepping stones. Offline planning trials are run with and without the steppability term included in the graph search, and the results are shown in Figure 4.



(a) Results for experience accumulation without steppability heuristic.



(b) Results for experience accumulation with steppability heuristic.

Fig. 4: Results for offline experience accumulation. Left plots showcase the results of all attempted subproblems – success, failure, or not attempted due to early trial termination – along with the total costs of the successful paths. Right plots show graph search times for each planning trial as well as average, minimum, and maximum TO solve times across all of the attempted subproblems within each planning trial. Note the difference in scale of the x-axes between Figure 4a and 4b.

Without the steppability term (Figure 4a), the planner must ascertain the obstacle locations through the experience heuristic. This can be viewed as as a form of intrinsic perception where the graph search only comes to avoid the obstacles through the high trajectory costs due to collisions being integrated into the edge weights. After 40 offline trials, the planner converges to a reliably successful contact path.

With the steppability term (Figure 4b), the planner has a weighting term on the graph edges that can act as a form of extrinsic perception. Based on the current view of the environment, this steppability term guides the graph search away from footholds that overlap with obstacles. The impact of this term can be seen in that the planner can immediately identify successful contact sequences. Furthermore, the only failure (Trial 4) was due to a kinematically infeasible transition, not a collision.

With the steppability term included, the graph search and trajectory optimization times increase significantly, but this is largely because the planner must be run in a simulation environment in order to read the incoming depth image stream. Without the steppability term, the experience accumulation process can be run outside of a simulation environment. Both with and without the steppability term, the TO solve times become more consistent once a collision-free contact sequence has been identifed by the graph search.

B. Online Trajectory Tracking

The reference trajectories generated from the contact planner are then passed to a nonlinear MPC-based tracking controller. We showcase the kinodynamic feasibility of the resulting reference trajectories in Figure 5.

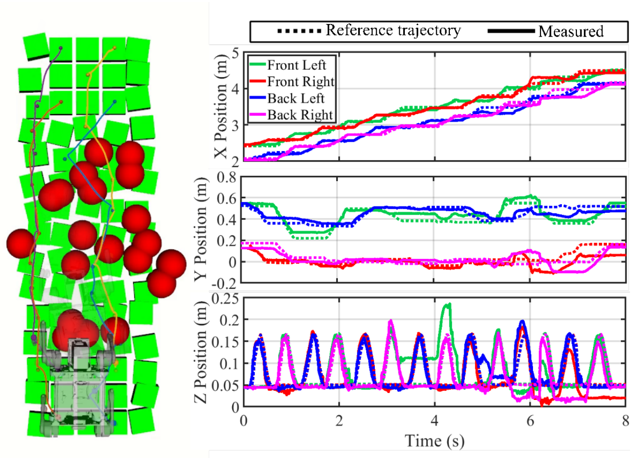


Fig. 5: Online tracking performance of the offline reference trajectory in simulation.

Overall, the tracking performance in simulation is suitable for online deployment, and the reference trajectory successfully guides the platform the other side of the stepping stones layout. At roughly 6 seconds, the Z-coordinates of the front right and back right feet fall below 0.05 m. This is due to the feet momentarily slipping off the desired stepping stones.

IV. Conclusion

In the future, we aim to validate the performance of our perception-informed contact planner on hardware in the real world. To do so, we plan on generating synthetic data from higher fidelity simulation environments and bridging the sim-to-real gap by incorporating a domain alignment process that merges simulation and real world data distributions by both corrupting simulation data and denoising real world data. We also seek to benchmark our steppability model against classical point cloud processing-based representations.

References

[1] L. D. Jackel, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "The DARPA LAGR program: Goals, challenges, methodology, and phase I results," *Journal of Field Robotics*, vol. 23, no. 11-12, pp. 945–973, 2006.

[2] J. Pippine, D. Hackett, and A. Watson, "An overview of the Defense Advanced Research Projects Agency's Learning Locomotion program," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 141–144, Feb. 2011.

[3] L. Jackel, D. Hackett, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "How DARPA structures its robotics programs to improve locomotion and navigation," *Communications of the ACM*, vol. 50, no. 11, pp. 55–59, Nov. 2007.

[4] M. Bajracharya, A. Howard, L. H. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the LAGR program," *Journal of Field Robotics*, vol. 26, no. 1, pp. 3–25, 2009.

[5] Michael Shneier, Will Shackleford, Tsai Hong, and Tommy Chang, "Performance Evaluation of a Terrain Traversability Learning Algorithm in the DARPA LAGR Program," Technical Report, 2008.

[6] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness, "Towards learned traversability for robot navigation: From underfoot to the far field," *Journal of Field Robotics*, vol. 23, no. 11-12, pp. 1005–1017, 2006.

[7] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, Feb. 2011.

[8] P. D. Neuhaus, J. E. Pratt, and M. J. Johnson, "Comprehensive summary of the Institute for Human and Machine Cognition's experience with LittleDog," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 216–235, Feb. 2011.

[9] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, Feb. 2011.

[10] A. Rankin, M. Bajracharya, A. Huertas, A. Howard, B. Moghaddam, S. Brennan, A. Ansar, B. Tang, M. Turmon, and L. Matthies, "Stereo-vision-based perception capabilities developed during the Robotics Collaborative Technology Alliances program," G. R. Gerhart, D. W. Gage, and C. M. Shoemaker, Eds., Orlando, Florida, Apr. 2010.

[11] J. Seo, S. Sim, and I. Shim, "Learning Off-Road Terrain Traversability With Self-Supervisions Only," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4617–4624, Aug. 2023.

[12] G. Vecchio, S. Palazzo, D. C. Guastella, D. Giordano, G. Muscato, and C. Spampinato, "Terrain traversability prediction through self-supervised learning and unsupervised domain adaptation on synthetic data," *Autonomous Robots*, vol. 48, no. 2, p. 4, Mar. 2024.

[13] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive Locomotion through Nonlinear Model Predictive Control," in *arXiv*, Aug. 2022.

[14] A. Bratta, A. Meduri, M. Focchi, L. Righetti, and C. Semini, "ContactNet: Online Multi-Contact Planning for Acyclic Legged Robot Locomotion," in *arXiv*, Mar. 2023.

[15] L. Wellhausen and M. Hutter, "ArtPlanner: Robust Legged Robot Navigation in the Field," *Field Robotics*, vol. 3, no. 1, pp. 413–434, 2023.

[16] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where Should I Walk? Predicting Terrain Properties From Images Via Self-Supervised Learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1509–1516, Apr. 2019.

[17] D. Driess, J.-S. Ha, and M. Toussaint, "Deep Visual Reasoning: Learning to Predict Action Sequences for Task and Motion Planning from an Initial Scene Image," in *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, Jul. 2020.

[18] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep Visual Heuristics: Learning Feasibility of Mixed-Integer Programs for Manipulation Planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 9563–9569.

[19] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical Planning for Long-Horizon Manipulation with Geometric and Symbolic Scene Graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 6541–6548.

[20] M. Nieuwenhuisen, J. Stueckler, A. Berner, R. Klein, and S. Behnke, "Shape-Primitive Based Object Recognition and Grasping," in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012, pp. 1–5.

[21] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 494–10 501.

[22] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," Aug. 2018.

[23] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, "Detectron2," 2019.

[24] M. Asselmeier, J. Ivanova, Z. Zhou, P. A. Vela, and Y. Zhao, "Hierarchical Experience-informed Navigation for Multi-modal Quadrupedal Rebar Grid Traversal," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[25] Z. Kingston and L. E. Kavraki, "Scaling Multimodal Planning: Using Experience and Informing Discrete Search," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 128–146, Feb. 2023.