End-to-end Differentiable Model of Robot-terrain Interactions

Ruslan Agishev [1 2] **Tomáš Tichý** [1] **Vladimír Kubelka** [2] **Martin Pecka** [1] **Patrik Vacek** [1] **Tomáš Svoboda** [1]
Karel Zimmermann [1]

Abstract

We propose a differentiable model of robot-terrain interactions that delivers the expected robot trajectory given an onboard camera image and the robot control. The model is trained on a real dataset that covers various terrains ranging from vegetation to man-made obstacles. Since robot-endangering interactions are naturally absent in real-world training data, the consequent learning of the model suffers from *training/testing distribution mismatch*, and the quality of the result strongly depends on generalization of the model. Consequently, we propose a grey-box, explainable, physics-aware, and end-to-end differentiable model that achieves better generalization through strong geometrical and physical priors. Our model, which functions as an image-conditioned differentiable simulation, can generate millions of trajectories per second and provides interpretable intermediate outputs that enable efficient self-supervision. Our experimental evaluation demonstrates that the model outperforms state-of-the-art methods.

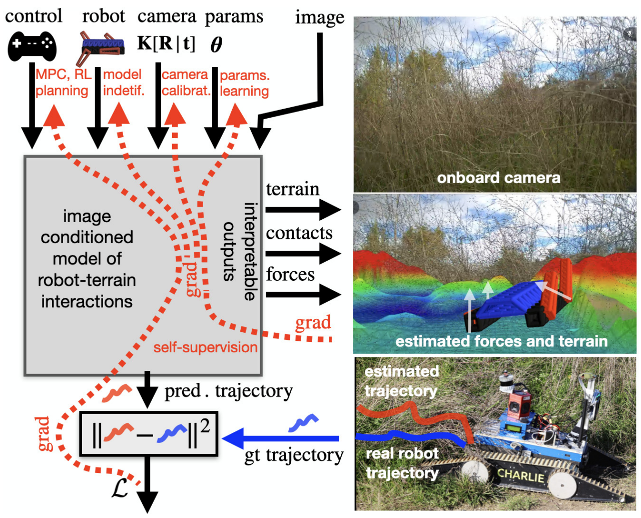


Figure 1. **Model overview:** The proposed model can be seen as an image-conditioned differentiable simulation that delivers a million simulated trajectories per second on the terrain depicted in the onboard camera image. The explainable structure also delivers many intermediate interpretable outputs that can serve for efficient self-supervision.

1. Introduction

Reliable navigation in a wild, unstructured environment remains the key challenge in deploying autonomous robotic platforms during real-world missions. The main difficulty stems from the inability to train an accurate model that predicts the outcome of robot-terrain interactions in robot-endangering situations. The reason is that the training inherently suffers from a severe *training/testing distribution mismatch* since robot-endangering and robot-devastating situations are naturally absent in real-world training data. Consequently, the generalization on these dangerous out-of-training-distribution situations is imperative for any real-world deployment.

In order to achieve generalization, roboticists proposed a wide variety of *white-box* [1]–[3] and *black-box* models [4]–[8]. While *white-box* models suffer from oversimplifications and inability to adapt to a new domain without massive hand tuning, *black-box* models suffer from poor generalization, weak explainability, and the need to gather expensive training data when re-trained. In contrast, we introduce a *grey-box*, explainable, physics-aware, and end-to-end differentiable model that enables self-supervised learning.

We focus on the problem of predicting the robot's trajectory given a single image captured by an onboard camera. In contrast to common black-box models, the proposed architecture comprises strong geometrical and physical priors that yield superior generalization. The resulting model thus feels like *learnable physics engine conditioned by a real image* that delivers *one million trajectories per second*; see Figure 1 for details. In addition to that, the model is

[1]VRAS group, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic [2]RNP Lab of the AASS Research Centre, Örebro University, Örebro, Sweden. Correspondence to: Ruslan Agishev <agishrus@fel.cvut.cz>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

end-to-end-differentiable; therefore, gradients can be backpropagated towards its (i) convolutional filters, (ii) camera and robot parameters, and (iii) control. The differentiability, in conjunction with the rapid simulation speed, draws the model suitable for a myriad of tasks, including model predictive control [9], trajectory shooting [10], supervised and reinforcement learning [11], online robot-model reidentification or camera recalibration [12]. The explainable structure of the proposed architecture also delivers a variety of intermediate outputs, such as terrain shape and its physical properties, robot-terrain reaction forces or contact points, which can all serve as efficient sources of self-supervision if measured during the training set creation.

In particular, we employ self-supervision from the robot's trajectories estimated by a common SLAM procedure, geometrical heightmaps estimated from lidar scans [5] and material types estimated through Microsoft's image foundation model [13]. While lidar scans serve as an upper bound on the shape of predicted heightmaps, the image foundation model delivers prior explicit knowledge about the rigidity of some objects that cannot be traversed.

We propose several differentiable physics engines that convert the predicted terrain and control into trajectory. The first implementation is based on a simple kinematic model which assumes that the robot always lies at the minimum of its potential energy. Since this model contains a non-convex constrained optimization problem in the feedforward pass, we backpropagate it through its KKT conditions. [14]. The remaining three implementations explicitly model physics interaction between the robot body and non-rigid terrain. [15]. Backpropagation is based on auto-differentiation [16], Neural ODE [17] and Nvidia's WARP [18]. **Our main contributions** are as follows.

Image-conditioned simulation: The end-to-end differentiable image-conditioned simulation that predicts a million trajectories per second is suitable for a myriad of underlying tasks such as model predictive control, trajectory shooting, supervised and reinforcement learning, online robot-model reidentification or camera recalibration.

Self-supervised learning: Explainability of the proposed grey-box model provides several well-interpretable intermediate outputs that serve as a natural source of self-supervision.

Experimental evaluation on non-rigid terrains: The proposed model outperforms other state-of-the-art methods on non-rigid terrains, such as grass or soft undergrowth that deforms when traversed by the robot.

2. THEORY

A detailed overview of the proposed architecture that converts images and control commands into trajectories is depicted in Figure 2. The model consists of several learnable modules that deeply interact with each other. The *geometry module* carefully transforms visual features from input image into the heightmap space using known camera geometry. The *terrain encoder* further refines visual features into interpretable physical quantities that capture properties of the terrain such as its shape, friction, stiffness and damping. Next, the *force encoder* combines the terrain properties with the robot model, robot state and control commands and delivers reaction forces at points of robot-terrain contacts. Finally, the *physics engine* solves the equations of motion dynamics by integrating these forces and delivers the trajectory of the robot. Since the complete computational graph of the feedforward pass is retained, the backpropagation from an arbitrary loss, constructed on top of delivered trajectories, or any other intermediate outputs, is at hand.

2.1. Model architecture

Given an input image $\mathbf{z}$, the proposed architecture successively estimates geometric heightmap $\mathcal{H}_g$, terrain heightmap $\mathcal{H}_t$, robot-terrain forces $\mathbf{f}_i$ and trajectories τ. The geometrical map $\mathcal{H}_g$ is a multichannel 2D array whose first channel contains heights of the environment observed in the camera, and the remaining channels contain visual features. Similarly, the terrain map $\mathcal{H}_t$ is a multichannel 2D array whose first channel contains the heights at which terrain is assumed to start generating forces against the robot, and the remaining channels again contain visual features. The intuition is that $\mathcal{H}_t$ models a partially flexible layer of terrain (e.g. mud) that is hidden under fully flexible vegetation. It is estimated by subtracting the estimated heightmap decrease $\mathcal{H}_d$ from the geometrical heightmap $\mathcal{H}_g$. Both maps are augmented with visual features that are successively converted into terrain properties. The part of the architecture that predicts terrain properties is called *terrain encoder*. Given the predicted terrain properties, state of the robot and control commands (e.g. track speed or flipper position), the forces $\mathbf{f}_i$ acting on the robot are computed. Finally, the physics engine solves the robot motion equation and estimates the trajectory corresponding to the delivered forces.

2.2. Self-supervised learning

Self-supervised learning of the proposed architecture minimizes three different losses:

Trajectory loss $\mathcal{L}_\tau = \|\tau - \tau^\star\|^2$ that minimizes the difference between SLAM-reconstructed trajectory $\tau^\star$ and predicted trajectory τ.

Geometrical loss $\mathcal{L}_g = \|\mathbf{W}_g \circ (\mathcal{H}_g - \mathcal{H}_g^\star)\|^2$ that minimizes

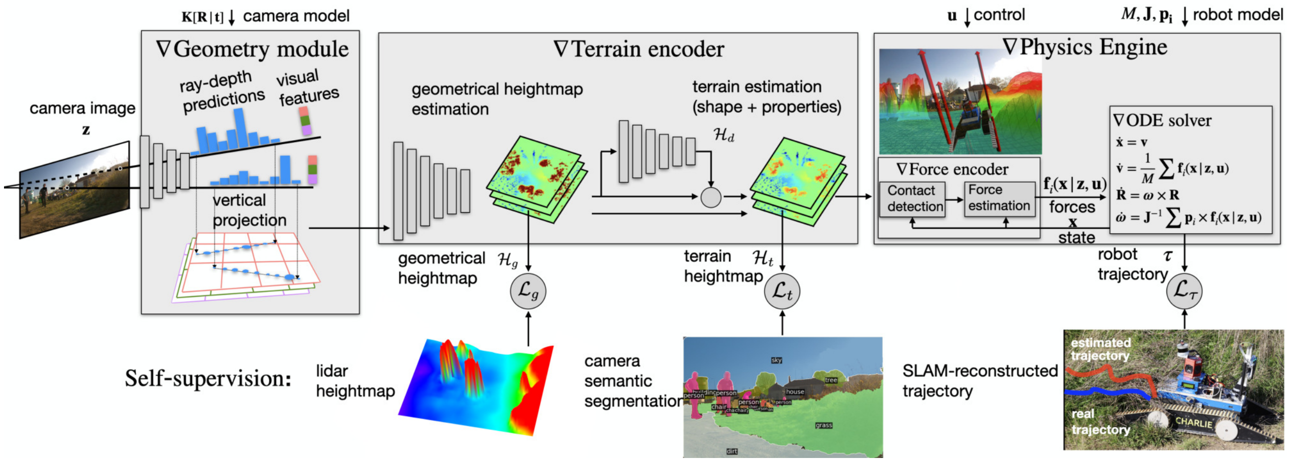

K[R|t] camera model
∇Geometry module
camera image z
ray-depth predictions
visual features
vertical projection
∇Terrain encoder
geometrical heightmap estimation
terrain estimation (shape + properties)
$\mathcal{H}_d$
geometrical heightmap $\mathcal{H}_g$
terrain heightmap $\mathcal{H}_t$
$\mathcal{L}_g$
$\mathcal{L}_t$
u control
$M, \mathbf{J}, \mathbf{p}_i$ robot model
∇Physics Engine
∇ODE solver
$\dot{\mathbf{x}} = \mathbf{v}$
$\dot{\mathbf{v}} = \frac{1}{M}\sum \mathbf{f}_i(\mathbf{x}|\mathbf{z},\mathbf{u})$
$\dot{\mathbf{R}} = \boldsymbol{\omega} \times \mathbf{R}$
$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\sum \mathbf{p}_i \times \mathbf{f}_i(\mathbf{x}|\mathbf{z},\mathbf{u})$
∇Force encoder
Contact detection
Force estimation
$\mathbf{f}_i(\mathbf{x}|\mathbf{z},\mathbf{u})$ forces
x state
robot trajectory τ
$\mathcal{L}_\tau$
Self-supervision:
lidar heightmap
camera semantic segmentation
sky
tree
person
house
grass
dirt
SLAM-reconstructed trajectory
estimated trajectory
real trajectory
CHARLIE

Figure 2. **Detailed architecture overview:** Neural network estimates depth predictions and rich visual features for each pixel ray. Depth-weighted visual features are vertically projected on 2.5D representation, and a geometrical heightmap $\mathcal{H}_g$ is estimated. This heightmap is further refined through *Terrain encoder*. It delivers terrain properties such as the heights of the rigid layer of terrain hidden under the vegetation, $\mathcal{H}_t = \mathcal{H}_g - \mathcal{H}_d$, or stiffness and dampening. Given state, control and terrain properties, forces at robot-terrain contacts are estimated. Finally, ∇*Physics engine* integrates these forces to estimate the resulting robot trajectory. Learning employs three losses: Trajectory loss, which measures the distance between the predicted and real trajectory; Geometrical loss, which measures the distance between the predicted geometrical heightmap and lidar-estimated heightmap; Terrain loss, which enforces rigid terrain on rigid semantic classes revealed through image foundation model.

the difference between ground truth lidar-reconstructed heightmap $\mathcal{H}_g^\star$ and predicted geometrical heightmap $\mathcal{H}_g$. $\mathbf{W}_g$ denotes an array selecting the heightmap channel corresponding to the terrain shape.

Terrain loss $\mathcal{L}_t = \|\mathbf{W}_t \circ (\mathcal{H}_t - \mathcal{H}_t^\star)\|^2$ that minimizes the difference between ground truth $\mathcal{H}_t^\star$ and predicted $\mathcal{H}_t$ terrain heightmaps for rigid objects detected through Microsoft's image segmentation model SEEM [19], that is derived from Segment Anything foundation model [13]. $\mathbf{W}_t$ denotes the array selecting heightmap cells that are covered by rigid materials (e.g. stones, walls, trunks), and $\circ$ is element-wise multiplication.

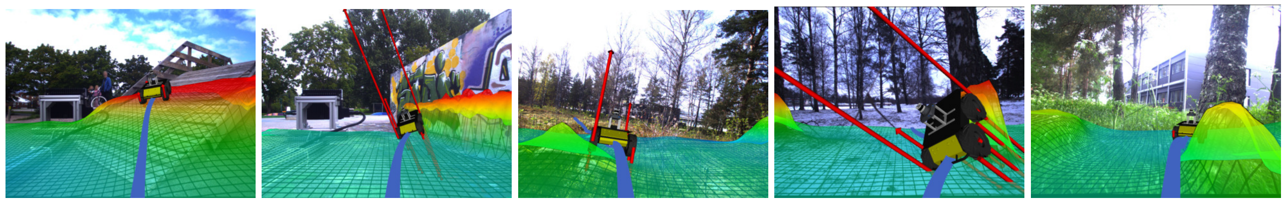
2.3. Implementation of differentiable physics engine

We implemented four physics engines that convert the terrain and control into trajectories. The first implementation is based on a simple kinematic model assuming that the robot always lies in the minimum of its potential energy [14]. The remaining three implementations explicitly model physical interactions between the robot body and non-rigid terrain [15].

Simple kinematic model assumes that the robot always lies in the minimum of its potential energy. Consequently, the feedforward pass contains a non-convex, constrained optimization problem. Backpropagation from ground truth trajectories is performed by constructing its KKT conditions. In particular, for each pose in the ground truth trajectory, we construct necessary conditions that make it an optimal solution to the optimization problem and then train the network to predict the terrain that satisfies these conditions. The resulting loss is called KKT-loss and it is detailed in [14]. All three following implementations explicitly model forces that appear during interaction between the robot and non-rigid terrain. These forces are then double-integrated into the final trajectory.

Auto-differentiation implementation uses Euler integrator in the feedforward pass and estimates gradient through the auto-differentiation [16], i.e. it builds and retains the full computational graph of the feedforward integration. The resulting implementation achieves only real-time speed; therefore, it is unsuitable for fast learning. In order to overcome this difficulty, we backpropagate from trajectories to terrain heightmaps before the training procedure and merge the resulting heightmaps with the ground truth terrain labels $\mathcal{H}_t^\star$. This simplification may fail in cases where multiple different terrains are consistent with a ground truth trajectory. We observed that the terrain ambiguity could be prevented by using sufficiently large heighmap bins.

Neural ODE implementation uses Runge-Kutta ODE solver in the feedforward pass and estimates the gradient through implicit function theorem as suggested in Neural ODE framework [17]. The resulting implementation is more accurate, especially when simulating almost non-penetrating contacts requiring huge forces. On the other hand, it runs about $0.2\times$real-time, making it intractable for large-scale learning.



(a) Moving up the ramp (b) Crashing into the wall (c) Overcoming a stone (d) Crashing into a tree (e) Traversing high grass

Figure 3. Robot-terrain interaction predictions in a diverse set of environments. The predicted heightmap $\mathcal{H}_t$, predicted robot's trajectories and interaction forces at contact points are being projected into RGB image plane. The heightmap color denotes its height (*blue* - low, *red* - high). *Note*: images (a) and (b) contain part of a robot's payload captured on the left side.

Nvidia's WARP implementation defines the physics engine kernel in Nvidia's WARP [18]. Since WARP allows parallel processing of multiple trajectories, the resulting implementation allows to generate 10^3 trajectories (10sec-long) in 0.3sec on GTX 1660 Ti GPU. Such speed is sufficient for large-scale learning which directly backpropagates from trajectories into the network kernels. Note that we found most of WARP's advertised non-core functionality, such as implemented physics of common geometric shapes, unusable due to many missing features, such as collision detection for cylinders. Consequently, we used only WARP's kernel compiler and implemented the physics from scratch.

3. EXPERIMENTS AND RESULTS

To evaluate the robot-terrain interaction models, we collected a dataset called RobInGas. It contains examples of robots moving in different weather conditions over hills, obstacles, and traversing high grass. It is recorded with the *tracked* (Figure 1) and *wheeled* (Figure 2.3) robotic platforms. The dataset contains point cloud scans from Ouster OS0-128, OS0-32 lidars and corresponding RGB images from Basler and IDS cameras installed on the robots. For each lidar-images data sample, we additionally record robot's poses. The localization was performed using SLAM methods [20], [21].

Figure 2.3 demonstrates *qualitative* results achieved on different types of terrains. All trajectories are predicted only from a single camera image and expected robot control. Observe that the resulting model reliably predicts robot-terrain interaction in dangerous scenarios that have not been covered by the training data. Examples in Figure 2.3 (a) and (b) demonstrate robot's motion on a rigid terrain. The red arrows represent normal components of the interaction forces that affect the robot's contact points. Note that the force values are much higher for the collision cases (b) and (d). Figure 2.3 (c) and (e) contain examples of a terrain with both rigid (tree trunks, stones) and traversable (grass) objects. The terrain encoder is able to differentiate between the two terrain types correctly for the most part of the images.

The strong generalization stems from the internal physics engine. Table 1 provides *quantitative* comparison on real trajectories with respect to state-of-the-art competitors. The vast majority of existing models estimate terrain properties and then use a simple kinematic model to infer the robot-terrain interaction. The resulting interaction is often limited to a binary decision about the terrain traversability or expected static pose of the robot on the terrain. To achieve a fair comparison, we employ our physics engine to convert predicted terrain properties on the trajectories for all competing models. Note that the dataset does not contain robot-endangering trajectories, on which the model outperforms the competitors the most due to its generalization. The codes and data are publicly available[1].

Table 1. Trajectory estimation accuracy on RobInGas data. ∇phys (Sec. 2.3) is used to predict trajectories τ (horizon $T = 10\,\text{sec}$).

input	**terrain encoder**	Δ**x** [m]	Δ**R** [°]
lidar	$\mathcal{H}_g$ interp. [22]	0.29	4.56
lidar	$\mathcal{H}_g$ pred. [14]	0.14	4.47
camera	$\mathcal{H}_g$, LSS [23]	0.11	5.53
camera	$\mathcal{H}_t$, **our**	**0.08**	**2.72**

4. CONCLUSIONS

We have proposed a physics-aware end-to-end differentiable model of robot-terrain interaction that predicts robot trajectory given the controls and onboard camera image depicting terrain in front of the robot. We studied four implementations of underlying physics engines: backpropagation through KKT stability conditions, Neural ODE, auto-differentiation, and Nvidia WARP. The resulting model may serve the robotics community as its rapid speed and differentiability make it suitable for large-scale learning and online planning and control.

[1] `https://github.com/ctu-vras/monoforce`

Acknowledgments

This work was co-funded by the Grant Agency of the CTU in Prague under Project SGS24/096/OHK3/2T/13, the Europen Union under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22_008/0004590), the Czech Science Foundation under Project 24-12360S, and the European Union's Horizon Europe Framework Programme under the euRobin project (ID: 101070596).

References

[1] S. Fabian, S. Kohlbrecher, and O. Von Stryk, "Pose prediction for mobile ground robots in uneven terrain based on difference of heightmaps," in *2020 IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR)*, 2020, pp. 49–56. DOI: 10.1109/SSRR50563.2020.9292574.

[2] S. Dogru and L. Marques, "An improved kinematic model for skid-steered wheeled platforms," *Autonomous Robots*, vol. 45, no. 2, pp. 229–243, 2021. DOI: 10.1007/s10514-020-09959-0.

[3] A. Manoharan, A. Sharma, H. Belsare, K. Pal, K. M. Krishna, and A. K. Singh, "Bi-level trajectory optimization on uneven terrains with differentiable wheel-terrain interaction model," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) under review*, 2024, pp. 1–8.

[4] A. Loquercio, A. Kumar, and J. Malik, "Learning visual locomotion with cross-modal supervision," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 7295–7302.

[5] C. Niu, C. Newlands, K.-P. Zauner, and D. Tarapore, "An embarrassingly simple approach for visual navigation of forest environments," *Frontiers in Robotics and AI*, vol. 10, 2023, ISSN: 2296-9144. DOI: 10.3389/frobt.2023.1086798.

[6] 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, 2019. DOI: 10.1109/LRA.2019.2895390.

[7] M. Guaman Castro, S. Triest, W. Wang, *et al.*, "How does it feel? self-supervised costmap learning for off-road vehicle traversability," in *ICRA*, 2023.

[8] G. Kahn, P. Abbeel, and S. Levine, "BADGR: an autonomous self-supervised learning-based navigation system," *CoRR*, vol. abs/2002.05700, 2020. arXiv: 2002.05700. [Online]. Available: https://arxiv.org/abs/2002.05700.

[9] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[10] W. Zeng, W. Luo, S. Suo, *et al.*, "End-to-end interpretable neural motion planner," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8660–8669.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[12] J. Moravec and R. Sara, "Robust maximum-likelihood online lidar-to-camera calibration monitoring and refinement," in *Computer Vision Winter Workshop*, Feb. 2018.

[13] F. Li, H. Zhang, P. Sun, *et al.*, "Semantic-sam: Segment and recognize anything at any granularity," *arXiv preprint arXiv:2307.04767*, 2023.

[14] V. Šalanský, K. Zimmermann, T. Petříček, and T. Svoboda, "Pose consistency KKT-loss for weakly supervised learning of robot-terrain interaction model," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5477–5484, 2021. DOI: 10.1109/LRA.2021.3076957.

[15] R. Agishev, K. Zimmermann, V. Kubelka, M. Pecka, and T. Svoboda, "Monoforce: Self-supervised learning of physics-aware model for predicting robot-terrain interaction," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) under review*, 2024, pp. 1–8.

[16] A. Paszke, S. Gross, and F. e. a. Massa, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[17] M. N. Ricky T. Q. Chen Brandon Amos, "Learning neural event functions for ordinary differential equations," in *ICLR*, 2021.

[18] M. Macklin, *Warp: A high-performance python framework for gpu simulation and graphics*, https://github.com/nvidia/warp, NVIDIA GPU Technology Conference (GTC), Mar. 2024.

[19] X. Zou, J. Yang, H. Zhang, *et al.*, "Segment everything everywhere all at once," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=UHBrWeFWlL.

[20] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013, ISSN: 0929-5593. DOI: 10.1007/s10514-013-9327-2.

[21] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1 729 881 419 841 532, 2019.

[22] Y. Wang, H. Li, X. Ning, and Z. Shi, "A new interpolation method in mesh reconstruction from 3D point cloud," in *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, 2011, pp. 235–242.

[23] J. Philion and S. Fidler, "Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D," in *Proceedings of the European Conference on Computer Vision*, 2020.