Modeling Sensors in Sim-to-Real: A Report

Mingyuan Chi
Department of Mathmatics. ETH Zürich, Zürich, Switzerland
minchi@student.ethz.ch

***Abstract*—This paper delves into the vital role of sensor modeling in the Simulation-to-Reality (Sim2Real) transfer, crucial in artificial intelligence and robotics. While Sim2Real offers safety and cost benefits, it faces the ”reality gap” challenge due to discrepancies between simulations and real-world conditions.**

Focusing on sensors such as RGB cameras and depth sensors, which are essential for real-world interaction, we review various sensor models, their methodologies, and challenges. This paper outlines the current state of sensor modeling in Sim2Real, highlighting its significance and the need for precision. We also explore future research directions to improve sensor model accuracy and real-world applicability. The paper is structured into three sections: an introduction to Sim2Real, a review of sensor modeling, and a discussion on challenges and future perspectives in sensor modeling within Sim2Real.

***Index Terms*—Sim2Real, Reinforcement Learning, Sensor Modeling**

I. Introduction

Training artificial intelligence systems or robots in real-world environments poses potential dangers and incurs significant expenses. Simulations offer a safer and more cost-effective alternative, mitigating risks commonly associated with real-world experiments. In scenarios where real-world data is limited or challenging to collect, simulations provide a viable means to generate the necessary data, thereby facilitating comprehensive training and testing. Sim2Real refers to the process of transferring knowledge, skills, or models developed in a simulated environment (Sim) to real-world applications (Real). This approach bridges the gap between theoretical models and practical, real-world utility. A primary challenge in Sim2Real is the ”reality gap,” characterized by discrepancies between simulated and real-world conditions. These discrepancies may arise from differences in physics, sensor data, lighting, and other environmental factors. To address the reality gap, techniques such as Domain Randomization are employed. This method introduces variability into simulations to better approximate the unpredictability of the real world, thereby enhancing the robustness of Sim2Real transfers. [7] Another approach involves the use of high-fidelity simulations, which strive to enhance the realism of simulated environments. Improvements in physics modeling, material properties, lighting, and sensor behavior aim to more closely replicate real-world conditions. [2], [3], [5], [13], [24], [25]

Developing models that exhibit generalizability and robustness is also crucial. This includes implementing continuous learning strategies and improving model architectures to ensure effective Sim2Real applications. [1]

Transitioning from the broader context of Sim2Real challenges, this paper narrows its focus to a critical component: sensor modeling. Sensors, including RGB cameras and depth cameras, are the primary means through which a policy interacts with the real environment. Therefore, the precise modeling of these sensors is essential in the Sim2Real process. This paper aims to explore the complexities of sensor modeling within the Sim2Real framework, underscoring its importance and addressing the unique challenges it presents. Our discussion will contribute to the understanding of how sensor accuracy impacts the effectiveness of Sim2Real transfer, thereby offering insights into optimizing sensor models for real-world applications.

The structure of this article is as follows:

- **Introduction to Sim2Real** : The first section offers a comprehensive introduction to the concept of Sim2Real. It underscores the pivotal role of sensor modeling in bridging the gap between simulated environments and real-world applications, setting the stage for a deeper exploration of the subject.
- **Sensor Modeling in Sim2Real** : The second section provides an extensive review of various sensor models utilized in Sim2Real. It encompasses a range of sensor types, including RGB cameras, depth sensors, Inertial Measurement Units (IMUs), force sensors, and encoders. This section aims to dissect the methodologies, advantages, and limitations associated with each sensor type, offering insights into their practical applications in Sim2Real scenarios
- **Challenges and Future Directions** : The third section is dedicated to summarizing the prevailing challenges in sensor modeling within the Sim2Real context. It also outlines potential future research directions, aiming to address these challenges and enhance the efficacy of sensor models. This section aims to stimulate further research and development in the field, paving the way for more sophisticated and reliable Sim2Real transitions.

II. Review of Literature

A. RGB camera

RGB cameras, notable for their accessibility and affordability, offer a high-density information format through pixels. The standard approach, as Zhu et al. [32] illustrate, employs a **C**onvolutional **N**eural **N**etwork (CNN) to encode the RGB stream from the camera, followed by a **L**ong **S**hort-**T**erm **M**emory (LSTM) network to model the signal sequence.

Inspired by the impressive capabilities of **Ne**ural **R**adiance **F**ields (Nerf), Byravan [3] utilized NeRF to construct simulation environments from short mobile phone videos. Although it's easy to take a video of 4 5 mintes, this process, involving lengthy preprocessing with COLMAP[1] (3-4 hours) and NeRF training (20 minutes on 8 V100 GPUs), is resource-intensive. Additionally, it demands manual calibration of the NeRF mesh with the real world and struggles with dynamic objects, which are instead incorporated using the MuJoCo simulation environment. Figure 1 detailed illustrates the whole pipeline of this approach.

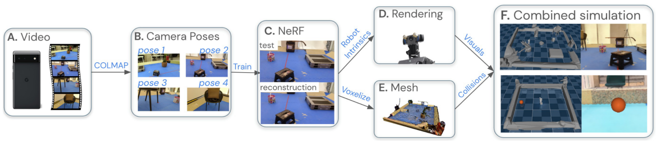

Fig. 1. The overview of pieline from Paper [3]. **A.**First collect a video using a generic phone, **B.** then use structure-from-motion software to label a subset of they video with camera poses. **C.** Third, a NeRF is trained on the labeled images, **D.** The image is rendered using the camera on the robot. **E.** And the well-trained Nerf is used to extract scene geometry as a mesh. **F.** Combine the rendered scene, mesh and dynamic objects together

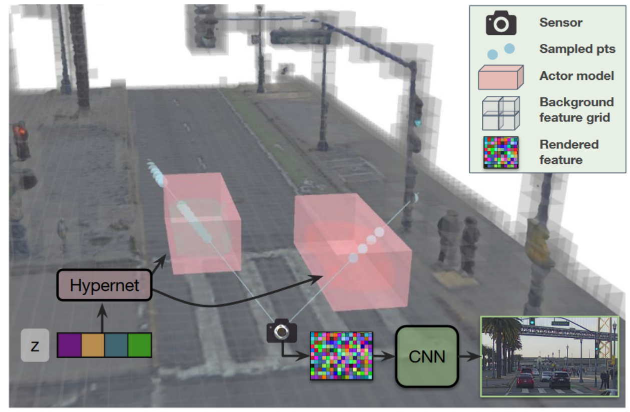

Fig. 2. Overview of the approach in Paper [28]. The 3D scene is divided into a static background (grey) and a set of dynamic actors (red). The static scene is modeled with a sparse feature-grid, and a hypernetwork is utilized to generate the representation of each actor from a learnable latent. The neural feature description is then produced through neural rendering.

Expanding upon the Neural Radiance Fields (NeRF) concept, Yang [28] introduced an innovative approach using voxel rendering, as opposed to traditional mesh rendering, to enhance memory efficiency and facilitate the handling of dynamic objects. In this methodology, a hypernetwork is employed to generate voxel-based representations for each dynamic actor within the simulated environment. This technique effectively manages the complexities of dynamic scene rendering, balancing detail and computational load.

To provide a clearer depiction of this technique, Figure 2 illustrates the voxel rendering process. As shown, the 3D scene is bifurcated into a static background (grey) and a set of dynamic actors (red). The static scene is represented through a sparse feature-grid, while the hypernetwork dynamically generates the voxel representation for each actor. This representation leverages a learnable latent space, and the scene is subsequently brought to life via neural rendering. This innovative approach, as visualized in the figure, underscores the efficiency and adaptability of voxel rendering in complex, dynamic environments.

Beyond environment creation, some research, like Yang's [29], focuses on generating realistic training images. This method involves a **Surf**ace **el**ement (Surfel)[2] GAN to transform surfel-rendered images into realistic RGB camera images, as described in Equation 1:

$$\text{Image(from Surfels)} \overset{\text{Surfel GAN}}{\leftrightarrow} \text{Images(from RGB cam)} \quad (1)$$

Although the surfel method can generate a more realistic dataset, it still suffers from problems like. a) SurfelGAN is unable to recover from broken geometry, b) Places where surfel map does not cover will cause Hallucination.

Addressing dataset augmentation, Lim, et al. [15] focus on **P**lanar **R**obot **C**asting (RPC) problem, combining both the simulation data and reality data as a large dataset to train the policy. It first lets the robot randomly interact with the environment to collect the reality dataset $\mathcal{D}_{\text{phy}} = \{\text{random real interaction}\}$. Then they sample a subset of the reality dataset to tune the robot parameters in the simulation $\theta_{\text{sim}} = \underset{\theta}{\text{argmin}}(s_{\text{real}}, s_{\text{sim},\theta})$. Third, they let the robot to randomly interact with environment in the simulation to get the simulation result $\mathcal{D}_{sim} = \{\text{random sim interaction}\}$. Then, they use the combined dataset to train the policy to obtain a better result $\pi = \text{Model}(\text{weighted combine}(\mathcal{D}_{\text{phy}}, \mathcal{D}_{\text{sim}}))$. However, Their work is hard to extend to 3D and inherent uncertainty about static and dynamic friction.

[1]COLMAP is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline

[2]A surfel is a small, oriented disk used to represent a portion of a 3D surface

B. Depth Sensor

Depth cameras, such as LiDAR, are instrumental in converting 2D RGB images to 3D spatial representations. LiDAR, leveraging laser time delay for distance measurement, is known for its reliability and precision, making it a popular choice in autonomous driving and robotics. Additionally, structured light methods, utilized in products like Kinect, Zed Camera, and Real Sense, offer alternative depth-sensing techniques. They are more flexible and accessible than LiDAR.

Although the depth from LiDAR is accurate compared to the structure light method, it still suffers from phenomena like a) Unretruned pulse because of fast amplitude decay $\frac{1}{R^4}$, b) Multiple echos caused by multiple surfaces, c) Spurious returns caused by beam divergence, d) Noisy points caused by ambiguity in waveform peak, and so on. Some work [18] uses methods like drop points, add points, spurious points, and noise points to bridge the reality gap in the simulation. Similarly, other work [6] use a Gaussian additive model and Bernoulli distribution approximation to model the noise model and point dropout. Their parameters are obtained through Equation 2 and Equation 4. And the parameters are assumed to obey the quadratic polynomial fit 35

$$\hat{d} \sim \mathcal{N}(d, \sigma) \quad (2)$$

$$\sigma = k_1\alpha^3 + k_2 d^2 + k_3 d\alpha + k_4\alpha + k_5 d + k_6 \tag{3}$$

$$p_r = \frac{np_t - np_a}{np_t} \tag{4}$$

$$p_r = p_1\alpha^2 + p_2 d^2 + p_3 d\alpha + p_4\alpha + p_5 d + p_6 \tag{5}$$

where $k_1 - k_6$ and $p_1 - p6$ are obtained by least squares fitting. d is the distance to the origin and α is the angle with forward axis. σ is the deviation and p_r is the Bernoulli distribution parameter. np_t, np_a are the theor etical/observed number of points.

Similar to SurfelGAN [29], CycleGAN is also used to generate more realistic simulation data [20]. It considers the problem from real LiDAR data to simulation LiDAR data as an image-to-image translation, therefore, it's viable to use a CycleGAN to learn this translation. Furthermore, not only CycleGAN could be used to do the data augmentation, **N**eural **S**tyle **T**ransfer model could used to augment the dataset from simulation. Sallab [21] provides a general framework to augment the LiDAR dataset with different methods including CycleGAN and NST. The overall architecture could be abstracted as Equation 6

$$\text{Realistic} = \mathcal{G}(\text{Simulation}, \text{Real}) \tag{6}$$

where the $\mathcal{G}$ could be CycleGAN in former work [20], and $\mathcal{G}$ is either CycleGAN or NST in latter work [21].

Apart from LiDAR, Kinect is also a typical choice for depth sensors. Mallick gives a detailed study on the noise model of Kinect [17]. And they also categorize the noise in the Kinect into three main classes, namely, spatial noise, temporal noise, and inference noise.

Chang [4] focuses on the DRC Plug task. Based on the Kinect RGBD image, they could restore the 3D position of the robotic arm. For the real world, they use a visual servoing approach, which uses the reverse Jacobian of the difference between PRE-INSERT frame and the "cable_tip" frame, to align the cable-tip pose with the socket pose. After each iteration, they will use the reality result to update the parameters in the simulation process, to be specific to the stiffness and damping matrix in the kinematics Equation 7.

$$\underset{K,D}{\text{argmin}} \| M\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + G + J^\top \mathbf{f}_{\text{ext}} + K\mathbf{q} + D\dot{\mathbf{q}} - \tau \| \tag{7}$$

where, K, D are the stiffness and damping matrix which are parameters for the simulation. M, C, G are matrices for inertia, centrifugal, Coriolis forces, and gravitational forces or torques. Therefore, their method is a closed loop for simulation and reality trials.

Tong [26] focus on the **D**eformable **L**inear **O**bjects problem. In the experiment, they use the Realsense camera to capture the 3D position of nodes on the linear object. Based on the geometry of the linear object and the target pattern, a policy network is trained in simulation and then is deployed to the real robot.

C. *Inertial Measurement* ***U****nit*

The **I**nertial **M**easurement **U**nit (IMU), primarily used for measuring acceleration, is often integrated with other sensors like encoders and RGBD sensors rather than being used standalone. Unlike RGBD sensors, IMUs generate signals at a significantly higher frequency, posing challenges in synchronizing these asynchronous signals, a topic still open for research.

A straightforward approach is to sample IMU signals at a specific frequency f, as explored by Işcen et al. [12]. They suggest directly feeding these sampled signals into the policy network and coupling them with motor position data for input to a **P**roportional-**D**erivative (PD) controller, thereby stabilizing the output.

In terms of signal processing, Gu [9] applies a Butterworth low-pass filter with a 15 Hz cutoff frequency to IMU data for denoising and outlier removal, with the IMU sampling frequency set at 30 Hz. Subsequently, a **L**ong **S**hort-**T**erm **M**emory (LSTM) controller is employed to interpret these time-sequenced signals.

Further, Imai et al. [11] adopt a multi-modal policy network, integrating inputs from RGB video, 4D IMU data (capturing roll and pitch angles and angular velocities), 12D robot joint rotation, and 12D of the last executed action. They address the asynchrony issue using a visual observation buffer and implement a **M**ulti-**M**odal **D**elay **R**andomization (MMDR) technique to enhance Sim2Real transfer.

Principal **C**omponent **A**nalysis (PCA) offers another method for interpreting IMU data, as demonstrated by Weerakoon et al. [27], who focus on trajectory navigation. By analyzing the first two principal components of IMU data, they extract surface-level vibration information. Additionally, they employ a **D**ynamic **W**indow **A**pproach (DWA) to penalize velocities and prevent robot flip-overs.

Lastly, IMUs can provide vital heading information, particularly when combined with **R**eal-**T**ime **K**inematic (RTK) GPS for precise localization. Zhang et al. [30] utilize an Extended Kalman Filter for velocity estimation in this setup. This facilitates the calculation of error states relative to a target reference trajectory, which can then be fed into a **M**odel **P**redictive **C**ontrol (MPC) controller for enhanced navigational accuracy.

D. *Force Sensor*

In their comprehensive review, Luo et al. [16] delineate the primary categories of force sensors utilized in robotic applications, classifying them into three distinct groups: single-point contact sensors, tactile sensor arrays, and optical tactile sensors. Single-point contact sensors, exemplified by devices such as the ATI Nano and biomimetic whiskers, are characterized by their high precision in measuring contact force and vibration, albeit limited to a singular point of contact. In contrast, tactile sensor arrays, which include technologies like fiber optics, MEMS-based barometers, and DigiTacts, trade-off precision for broader measurement capabilities. Optical tactile sensors, a burgeoning field of research, combine accuracy with high-density measurement capabilities, with notable examples including GelSight, GelTip, TacTip, and DIGIT.

The traditional way to model the tactile array is to use a numerical approach [14]. A jacobian could be calculated according to the tactile pattern signal from the tactile array. Different patterns correspond to different Jacobian. Figure [?] gives some examples of this. Then inverse of Jacobian could be fed backward to the controller system. The pipeline is numerical robust and fast. However, it's coarse and limited to specific contact configurations.

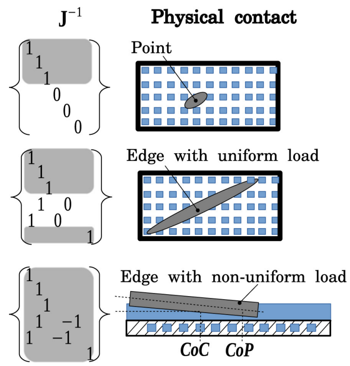

Fig. 3. The Inverse Jacobian Example for Tactile Array

In the context of specialized shape sensors like the SynTouch BioTach, Narang et al. [19] employs Two **V**ariational **A**uto **E**ncoders structure for modeling. The first autoencoder encodes the mesh deformation $m_t \in \mathbb{R}^{4k\times 3}$ from simulation into a latent space $z_m \in \mathbb{R}^{128}$. And the second one will encode the electrode signals $e_t \in \mathbb{R}^{19}$ from the sensor into another latent space $z_m \in \mathbb{R}^{8}$. Then two **F**ully **C**onnected **N**etworks are used to bridge the two latent spaces.

As for optical tactile, a simple deformation-rendering model can be used to describe how the GelSight work [8]. In this work, they first generate a Rough elastomer heightmap according to the contact region. Secondly, Gaussian filtering is applied to simulate the strain of the membrane. Finally, a simulation image could be rendered using Phong's model. The pipeline is expressed compactly in Equation 8.

$$\text{RGB} = \text{Phong}(\text{GF}(H_{\text{height map}})) \tag{8}$$

While this method is both rapid and robust in generating tactile sensor data, it is important to note that the rough elastomer heightmap, Gaussian filtering, and Phong's model collectively offer only an approximate representation. This approximation may lead to inaccuracies in the model, particularly in contexts where fine details of tactile interaction are crucial. To provide a clearer understanding of this process, Figure 4 illustrates the rendering pipeline as per Equation 8.

To better visualize the Equation 8, Figure 4 is shown here.

The **F**inite **E**lement **M**ethod provide a more precise modeling approach. Several studies have focused on bridging the gap between FEM simulation and Real-world indentations.

Sferrazza [23] directly utilize the simulation force distribution from FEM to train the model. Anothor significant work [22] combine the optical flow from FEM simulation with force distribution as a training dataset.

For specific tasks like determing the in-hand pose of tubar objects, Zhao [31] employs an AngleNet to extract the angle of the tubar from the image provided by DIGIT force sensor.

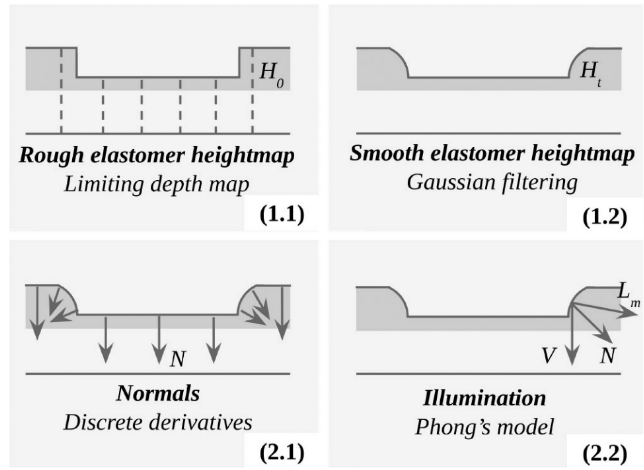

Fig. 4. The two steps of Equation 8. First elastomer heightmap is first approximated from the depth camera. Then, the image is smoothed using Gaussian filter and rendered utilizing Phong's illumination model.

The parameters of AngleNet are trained thorugh the simulation process in Gazebo. To enhance the corelation between simulated and real-world data, a CycleGAN-like architecture "CTF-CycleGAN" is utilized for transforming the simulated images into more realistic representations.

E. Encoders

Normally, encoders are electrical sensors inside the motor that turn the angular velocity or torque into electrical data. They have been widely used to get feedback since the early years. In most cases, encoders are used with other sensors like IMU and RGBD cameras. Unfortunately, there are only a few studies in modeling the encoders in the simulation.

In the simulation framework NeuronGym [10], the encoders are modeled using Gaussian noise model 9

$$\tilde{\omega}_i(t) = \omega_i(t) + n^e, n^e \sim \mathcal{N}(\mu_e, \sigma_e) \tag{9}$$

where μ_e and σ_e are the mean and standard deviation of the measurement noise.

III. Conclusion

In this paper, we have explored the critical importance of sensor modeling in the Sim2Real process, highlighting its benefits for safety, cost-effectiveness, and data availability in AI and robotics.

Our focused examination of sensor models, especially RGB cameras, depth sensors, IMU, force sensors, and encoders, revealed their significance in accurately bridging the gap between simulated and real environments.

From the studies, we found that,

- There are not many works modeling the encoders in sim2real.
- CycleGAN is widely used for learning the transition from simulation data to real-world data.

Therefore, future research in Sim2Real is encouraged to cover a wider range of sensors like encoders. Moreover, the robustness of the model is also an open topic. The advancements in this field will be instrumental in furthering the capabilities and efficiency of AI and robotic systems in real-world scenarios.

REFERENCES

[1] Saminda Wishwajith Abeyruwan, Laura Graesser, David B D'Ambrosio, Avi Singh, Anish Shankar, Alex Bewley, Deepali Jain, Krzysztof Marcin Choromanski, and Pannag R Sanketi. i-sim2real: Reinforcement learning of robotic policies in tight human-robot interaction loops. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 212–224. PMLR, 14–18 Dec 2023.

[2] Prithvi Akella, Wyatt Ubellacker, and Aaron D Ames. Safety-critical controller verification via sim2real gap quantification. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10539–10545. IEEE, 2023.

[3] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, et al. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9362–9369. IEEE, 2023.

[4] Peng Chang and Taşkin Padif. Sim2real2sim: Bridging the gap between simulation and real-world in flexible object manipulation. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pages 56–62, 2020.

[5] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021.

[6] João Espadinha, Ivan Lebedev, Luka Lukic, and Alexandre Bernardino. Lidar data noise models and methodology for sim-to-real domain generalization and adaptation in autonomous driving perception. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 797–803. IEEE, 2021.

[7] Haichuan Gao, Zhile Yang, Xin Su, Tian Tan, and Feng Chen. Adaptability preserving domain decomposition for stabilizing sim2real reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4403–4410. IEEE, 2020.

[8] Daniel Fernandes Gomes, Paolo Paoletti, and Shan Luo. Generation of gelsight tactile images for sim2real learning. *IEEE Robotics and Automation Letters*, 6(2):4177–4184, 2021.

[9] Jiacheng Gu and Zhibin Li. Learning autonomous mobility using real demonstration data. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 428–434. IEEE, 2021.

[10] Li Haoran, Liu Shasha, Ma Mingjun, Hu Guangzheng, Chen Yaran, and Zhao Dongbin. Neuronsgym: A hybrid framework and benchmark for robot tasks with sim2real policy learning. *arXiv preprint arXiv:2302.03385*, 2023.

[11] Chieko Sarah Imai, Minghao Zhang, Yuchen Zhang, Marcin Kierebiński, Ruihan Yang, Yuzhe Qin, and Xiaolong Wang. Vision-guided quadrupedal locomotion in the wild with multi-modal delay randomization. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5556–5563, 2022.

[12] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. In *Conference on Robot Learning*, pages 916–926. PMLR, 2018.

[13] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.

[14] Zhanat Kappassov, Juan-Antonio Corrales, and Véronique Perdereau. Touch driven controller and tactile features for physical interactions. *Robotics and Autonomous Systems*, 123:103332, 2020.

[15] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8282–8289, 2022.

[16] Shan Luo, Joao Bimbo, Ravinder Dahiya, and Hongbin Liu. Robotic tactile perception of object properties: A review. *Mechatronics*, 48:54–67, 2017.

[17] Tanwi Mallick, Partha Pratim Das, and Arun Kumar Majumdar. Characterizations of noise in kinect depth images: A review. *IEEE Sensors journal*, 14(6):1731–1740, 2014.

[18] Sivabalan Manivasagam, Ioan Andrei Bârsan, Jingkang Wang, Ze Yang, and Raquel Urtasun. Towards zero domain gap: A comprehensive study of realistic lidar simulation for autonomy testing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8272–8282, 2023.

[19] Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6444–6451. IEEE, 2021.

[20] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Nader Essam. Lidar sensor modeling and data augmentation with gans for autonomous driving. *arXiv preprint arXiv:1905.07290*, 2019.

[21] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Mohamed Shawky. Unsupervised neural sensor models for synthetic lidar data augmentation. *arXiv preprint arXiv:1911.10575*, 2019.

[22] Carmelo Sferrazza, Thomas Bi, and Raffaello D'Andrea. Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4389–4396. IEEE, 2020.

[23] Carmelo Sferrazza, Adam Wahlsten, Camill Trueeb, and Raffaello D'Andrea. Ground truth force distribution for learning-based tactile sensing: A finite element approach. *IEEE Access*, 7:173438–173449, 2019.

[24] Rohan P. singh, Zhaoming Xie, Pierre Gergondet, and Fumio Kanehiro. Learning bipedal walking for humanoids with current feedback. *IEEE Access*, 11:82013–82023, 2023.

[25] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[26] Dezhong Tong, Andrew Choi, Longhui Qin, Weicheng Huang, Jungseock Joo, and Mohammad Khalid Jawed. Sim2real neural controllers for physics-based robotic deployment of deformable linear objects. *The International Journal of Robotics Research*, page 02783649231214553, 2023.

[27] Kasun Weerakoon, Adarsh Jagan Sathyamoorthy, and Dinesh Manocha. Sim-to-real strategy for spatially aware robot navigation in uneven outdoor environments. *arXiv preprint arXiv:2205.09194*, 2022.

[28] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023.

[29] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] Harry Zhang, Stefan Caldararu, Sriram Ashokkumar, Ishaan Mahajan, Aaron Young, Alexis Ruiz, Huzaifa Unjhawala, Luning Bakke, and Dan Negrut. Zero-shot policy transferability for the control of a scale autonomous vehicle. *arXiv preprint arXiv:2309.09870*, 2023.

[31] Yongqiang Zhao, Xingshuo Jing, Kun Qian, Daniel Fernandes Gomes, and Shan Luo. Skill generalization of tubular object manipulation with tactile sensing and sim2real learning. *Robotics and Autonomous Systems*, 160:104321, 2023.

[32] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.