IndoorSim-to-OutdoorReal: Learning to Navigate Outdoors without any Outdoor Experience

Joanne Truong[1], April Zitkovich[2], Sonia Chernova[1], Dhruv Batra[1], Tingnan Zhang[2], Jie Tan[2], Wenhao Yu[2]
[1]Georgia Institute of Technology, [2]Google DeepMind

***Abstract*—We present IndoorSim-to-OutdoorReal (I2O), an end-to-end learned visual navigation approach, trained solely in simulated short-range indoor environments, and demonstrates zero-shot sim-to-real transfer to the outdoors for long-range navigation on the Spot robot. Our method uses zero real-world experience (indoor or outdoor), and requires the simulator to model no predominantly-outdoor phenomenon (sloped grounds, sidewalks, etc). The key to I2O transfer is in providing the robot with additional context of the environment (*i.e.* a satellite map, a rough sketch of a map by a human, etc.) to guide the robot's navigation in the real-world. The provided context-maps do not need to be accurate or complete– real-world obstacles (*e.g.* trees, bushes, pedestrians, etc.) are not drawn on the map, and openings are not aligned with where they are in the real-world. Crucially, these inaccurate context-maps provide a hint to the robot about a route to take to the goal. We find that our method that leverages Context-Maps is able to successfully navigate hundreds of meters in novel environments, avoiding novel obstacles on its path, to a distant goal without a single collision or human intervention. In comparison, policies without the additional context fail completely. We additionally find that the Context-Map policy is surprisingly robust to noise. In the presence of significantly inaccurate maps in simulation (corrupted with 50% noise, or entirely blank maps), the policy gracefully regresses to the behavior of a policy with no context. Videos are available on our project website.**

I. Introduction

Much of Earth's landmass outdoors is occupied by challenging terrain that is inaccessible to wheeled robots. On the other hand, humans and legged animals are able to explore most of this landmass by finding stable footholds to navigate through these challenging terrains. Legged platforms provide robust locomotion, and have demonstrated successful sim2real transfer in challenging and diverse terrain such as the outdoors [1]–[4]. Alongside these advancements, recent works in deep reinforcement learning have demonstrated success in training virtual robots to navigate efficiently in simulation before transferring the learned skills to real-world environments [5]–[13], on both wheeled and legged robots. These advances were made possible due to the development of fast, scalable simulators [10], [14]–[23] and the availability of large-scale datasets of photorealistic 3D scans of indoor environments [24]–[26]. In this paper, we seek to bring the same advancements made in indoor visual navigation to the outdoors for legged robots.

Outdoor navigation with legged robots is different from indoor navigation, and is relatively under-studied. While indoor navigation typically span tens of meters, outdoor navigation requires a robot to navigate hundreds of meters to travel from one building to another. At this scale of navigation, taking

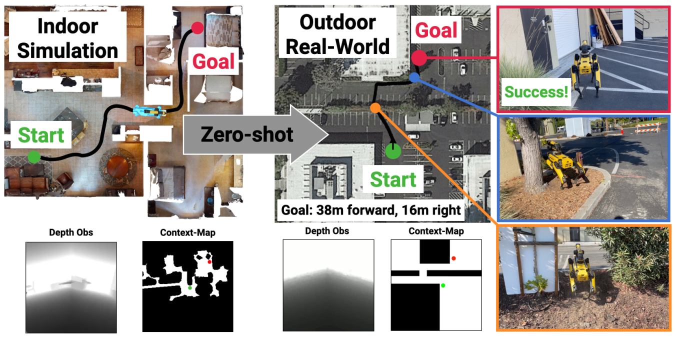

Fig. 1: We demonstrate zero-shot sim2real transfer for long-range outdoor navigation on the Spot robot. One key ingredient is to combine information from inaccurate high-level map (titled Context-Map) and accurate but limited onboard sensing. The robot leverages the map hint to navigate up the slope to the opening in the bushes (orange), takes a shortcut between a tree and a wooden pole (blue) that is not marked on the map but visible in egocentric images, to successfully reach the goal position a distant building (red).

the wrong turn or backtracking can be costly, and simple exploration methods can be inefficient.

We present IndoorSim-to-OutdoorReal (I2O), which enables a quadrupedal robot to successfully navigate hundreds of meters in novel outdoor environments, around previously unseen outdoor obstacles (trees, bushes, buildings, pedestrians, etc.), in different weather conditions (sunny, overcast, sunset) – despite being trained *solely in simulated indoor environments*. The robot has never seen any outdoor environments, and has only been trained using short-range trajectories ($\sim$ 8m).

We find that the key to enabling I2O is to provide the robot with additional context of its environment (*i.e.* via a satellite image, a rough sketch by a human, etc.), which allows the learned policy to bias its search, obviating the need for costly exhaustive search in the real-world. We provide additional context to the robot through a rough human sketch to guide the robot's navigation. These Context-Maps do not need to be accurate, but serve as a hint in the general directions that the robot should explore. These sketches enable a human user to specify obscure paths that may be difficult for the robot to find, or paths that are not visible in satellite images (*i.e.* a small opening through bushes shown in Figure 1). the robot must learn to use the maps in conjunction with observations from its onboard cameras to adapt to on-the-ground reality – avoiding obstacles not drawn on the map (bushes, chairs, people).

In our experiments, we find that the robot with the Context-Map is able to navigate hundreds of meters to a distant goal without a single collision or human intervention. In contrast, without the additional context, the robot completely fails at navigating long-ranges outdoors, and is only able to make minor progress towards the goal before getting stuck around obstacles. We conduct a comprehensive quantitative analysis in simulation, and demonstrate that in indoor environments, the additional context can improve success rate by 17%, and improve path efficiency by 22%. Additionally, we find that the Context-Map policy is surprisingly robust to noise in the provided Context-Map. When the maps are inaccurate (corrupted with 50% noise, or an entirely blank map), the performance of the policy gracefully regresses back to the performance of policies trained without any context.

II. IndoorSim-to-OutdoorReal Transfer

Task. In PointGoal Navigation (PointNav) [27], a robot is initialized in a novel environment and needs to navigate to a goal location. The robot has access to egocentric RGB-D observations, and an egomotion sensor. An episode is successful if the robot reaches within 0.425m of the goal.

Dataset. We use the Habitat-Matterport (HM3D) [24] and Gibson [28] 3D datasets, which consists of over 1000 scans of real-world indoor environments consisting of realistic clutter.

A. *Context-Guided PointGoal Navigation*

We aim to leverage context information freely accessible through public map services for long-range PointNav outdoors, which we denote as Context-Guided PointNav (ContextNav).

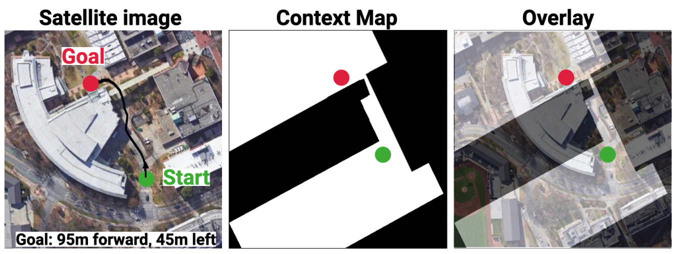

Fig. 2: Using a satellite image of the area from Google Maps (left), a human operator sketches a rough Context-Map for the robot (middle). Notice how defective the map is (right): large parts or entire buildings are missing, no roads or sidewalks are shown; but crucially, the map contains a hint for an opening to get to the goal.

We provide the robot with context in the form of an outdated map (Context-Map). The Context-Map input does not need to be very accurate, but should serve as a rough guide for general directions that the robot should explore (*i.e.* a satellite map showing roughly where buildings are). Consequently, the robot must use observations from its camera to adapt to novel obstacles or clutter present in the environment but absent on the map, and be willing to take shortcuts available in the world but shown as obstacles on the map. We represent the map as a top-down occupancy map, which can be obtained in the real-world by converting a satellite image to illustrate occupied and freespace through a human sketch, or through an automated process. Using a binary occupancy map provides a few benefits over directly using satellite images. An abstracted binary map allows the maps to be used for both indoor and

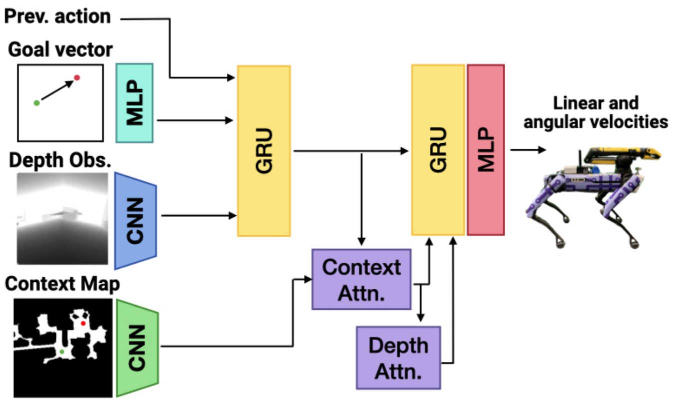

Fig. 3: The Context-Guided PointNav architecture. We use a CNN to process the Context-Map, and a GRU to process the attended features.

outdoor navigation, while satellite images are only applicable to outdoor navigation. Additionally the human operator can give hints to the robot about paths that may not be easily visible in the satellite image, or from the robot's initial position (*i.e.* openings in bushes). In our experiments, we use human-sketched Context-Maps, however these maps can easily be generated automatically by postprocessing a digital map.

Policy Architecture. We train a high-level visual navigation policy entirely in simulation using deep reinforcement learning. The No-Context policy architecture takes as input an egocentric depth image, a goal vector, and the action at the previous timestep. The goal vector is represented as the distance and heading relative to the robot's current pose. The output of the policy is the desired center-of-mass linear and angular velocities (v_x, ω) for the robot to follow. The depth observations are processed using a 3-layer CNN visual encoder, and the goal vector is encoded using a linear layer. These features are fed into a 1-layer GRU, followed by a single linear layer which parameterizes a Gaussian action distribution from which the action is sampled. Next, to incorporate additional context information for ContextNav we use freely accessible top-down maps of indoor environments in simulation. The Context-Map is represented as a $2 \times 100 \times 100$ matrix. The first channel of the map is an egocentric occupancy map, in which the cells denote obstacles (0) or freespace (1). The second channel of the map illustrates the agent's current location in the map and the location of the goal coordinate with a small disk (1), and 0 otherwise. We process the map context inputs using a ResNet18 visual encoder [29]. We compute the scaled dot-product attention [30] between the depth and context features, and pass the attended features into a second GRU.

B. *Techniques to Aid IndoorSim-to-OutdoorReal Transfer.*

Indoor-to-Outdoor Transfer. Two of the main differences between indoor and outdoor navigation are 1) navigation length (short vs. long), and 2) terrain type (flat vs. rocky/sloped). First, we found that traditional PointNav policies were highly sensitive to the goal vector. Since the policies were only trained in simulation, and typically see trajectories ~8m away, the policies failed to generalize to longer-range goals. We normalize the goal vector by using the log of the goal distance,

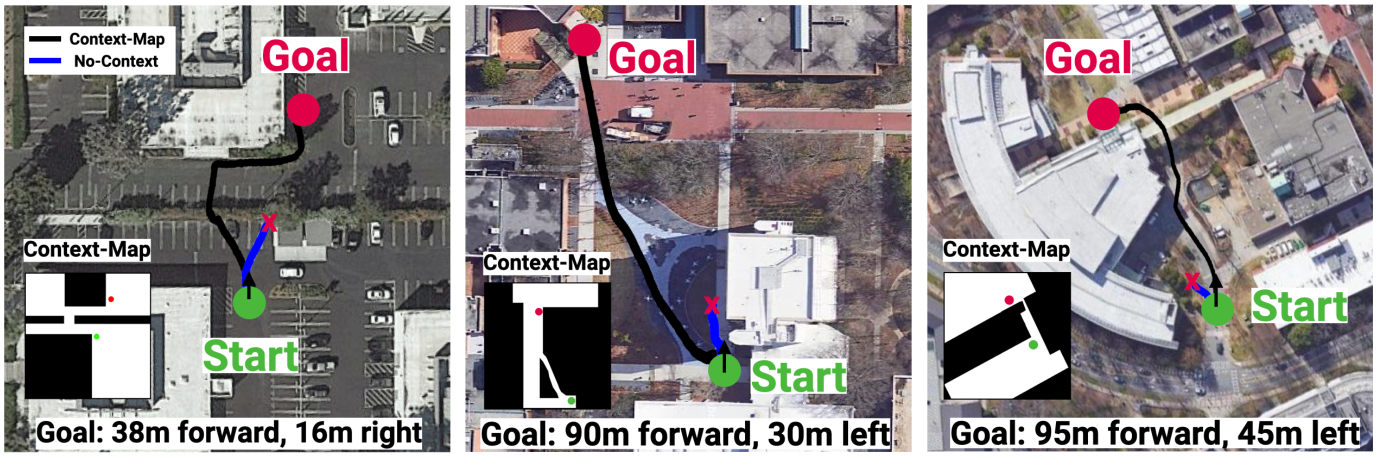
Context-Map
No-Context
Goal
Start
Context-Map
Goal: 38m forward, 16m right
Goal
Start
Context-Map
Goal: 90m forward, 30m left
Goal
Start
Context-Map
Goal: 95m forward, 45m left

Fig. 4: We test our policies in 3 novel environments in the real-world. The routes contain many obstacles including bushes, buildings, cars and pedestrians that the robot has never seen during training. The Context-Map policies (black) are able to navigate hundreds of meters to reach the goal 100% of the time. In comparison, the No-Context policies (blue) beeline towards the goal, resulting in episode failures.

which enabled longer-range navigation. Next, we found that naively trained PointNav policies had difficulty navigating up slopes. Since slopes are infrequent in indoor environments, slopes outdoors appear as a large obstacle in the robot's depth camera, and thus the robot avoids walking up the slope. To enable the robot to walk up and down slopes in the real-world, we artificially add slopes to the robot's observation by randomizing the pitch of the camera during training by $\pm 30°$.

Sim-to-Real Transfer. In simulation, instead of modeling the robot's low-level control, we use kinematic control as an approximation for the robot's movement. In kinematic control, the robot is moved to its next state via Euler integration at 2Hz without running full rigid-body physics. Kinematic control leads to better sim-to-real transfer through faster simulation, as compared to dynamic control [9]. Next, we filter the depth images from Spot's depth cameras in the real-world to better match observations from simulation. We use depth completion from [31] to fill in holes and smooth the image. Additionally, we set the pixels further away from the max depth range (3.5m) to white to match simulation. Lastly, we add Random Erasing noise [32] to our depth images during training to improve the robustness to missing pixels in the real-world.

III. Results

A. Zero-shot IndoorSim-to-OutdoorReal Navigation

In our experiments, we use the Boston Dynamics (BD) Spot robot. Our navigation policy outputs velocity commands, and we rely on BD's low-level controller for movement. We task the robot with navigating to 3 long-range goals outdoors (shown in Figure 4), with many real-world obstacles present (bushes, buildings, cars, tables, pedestrians, etc.).

No-Context Real-world Outdoor Navigation. First, we test to see if policies with No-Context can directly transfer to long-range navigation outdoors. The outdoor routes are complex and there does not exist a straight-line path to the goal; the robot is required to navigate around large obstacles to reach the goal successfully. We find that the No-Context policy immediately makes a beeline towards the goal using the goal vector for guidance. In indoor environments, there are more obstacles around the robot, such as walls, that guide the robot to avoid making a beeline to the goal. In the outdoors however, the robot makes a beeline presumably led by the depth sensors that indicate plenty of free-space around the robot. This leads the robot to wander into obstacles such as bushes, resulting in unsuccessful episodes. The No-Context policy completely fails to navigate in all three routes, each only making minor progress to the goal before an episode failure (Table I).

Route #	Goal	Method	SR ↑	Distance Travelled (m) ↑
1	38m Forward, 16m Right	No-Context	0.0	16.6$\pm$0.1
		Context-Map	**100.0**	**63.4**$\pm$2.5
2	90m Forward, 30m Left	No-Context	0.0	9.7$\pm$3.4
		Context-Map	**100.0**	**112.2**$\pm$1.8
3	95m Forward, 45m Left	No-Context	0.0	5.1$\pm$0.3
		Context-Map	**100.0**	**129.8**$\pm$2.8

TABLE I: We test the No-Context and Context-Map policies on 3 long-range outdoor routes (Figure 4).

Context-Guided Real-world Outdoor Navigation. Next, we test our Context-Map policy outdoors. We provide the robot with rough sketches indicating preferred paths for the robot to take for each route (Figure 4). Obstacles such as cars, trees, or chairs are not shown on the map, and only rough hints for an opening to the goal is depicted. We find that the Context-Map policy is able to leverage the Context-Maps to bias its search during outdoor navigation, and successfully reach the distant goal location 100% of the time (Table I), without a single collision or human intervention. With our approach, the robot was able to navigate around dynamic obstacles such as pedestrians despite these obstacles not being drawn directly on the Context-Map. Our approach is also not limited to 2D navigation; the robot navigates up slopes (Route 1), and can navigate in various terrains types. Our policies are also able to maintain a balance between the Context-Maps, and what it sees in its visual inputs. The robot leverages depth images to avoid clutter or dynamic obstacles, and Context-Maps are used for high-level guidance. While Context-Maps provide a means for the operator to specify a preferred route for the robot to take, the robot may take shortcuts along the way that are not present in the map when its vision senses freespace.

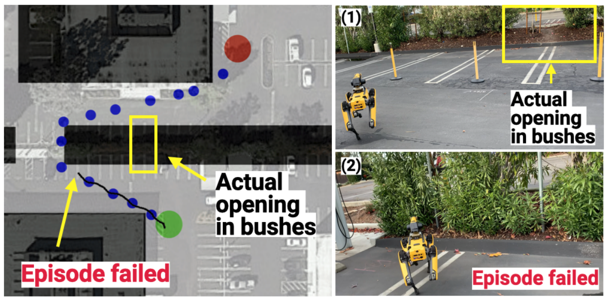

Fig. 5: **Left:** We run RRT* on an outdated Context-Map to find waypoints (blue) to the goal. We pass the waypoints to a policy to follow (black). **Right:** The waypoints lead the robot past the opening in the bushes (1), leading to a failure (2).

Traditional Planning Real-world Outdoor Navigation. Lastly, we test to see if classical approaches can be used for outdoor navigation by planning directly on the *exact same* outdated, inaccurate Context-Map. We run RRT* on the Context-Map to generate a list of waypoints to the goal. These waypoints are then used with a policy trained in simulation to follow waypoints along the shortest route to the goal. We find that the policy is able to successfully navigate to each successive waypoint. However, the robot ended up missing the actual opening in bushes because the waypoints were generated from an outdated map (shown in Figure 5). This confirms our conjecture that classical planning based approaches are highly sensitive to the map input, and are not able to adapt to inaccuracies. In the real-world, creating a perfect, and always-up-to-date map is not realistic.

B. Simulation Results

Indoor Navigation. We find that the Context-Map policies achieve high success rate and SPL (95.4 SR and 81.0 SPL; Table II, row 2), demonstrating that the policy is able to utilize its given Context-Map to navigate to the goal. In contrast, the No-Context policies achieve a lower success rate of 78.7% (-16.7% SR). We also see a +24.6% SPL between the two policies, demonstrating that the robot can navigate to the goal more efficiently using the added context. These experiments demonstrate that Context-Map policies are capable of complex navigation through multiple cluttered indoor rooms.

Method	SR ↑	SPL ↑
No-Context	$78.7_{\pm 8.2}$	$56.4_{\pm 4.4}$
Context-Map	$95.4_{\pm 0.2}$	$81.0_{\pm 2.4}$

TABLE II: The policy using Context-Maps outperforms the No-Context policy (+16.7% SR, +24.6% SPL).

Indoor Navigation using Outdated Maps. We test the robustness of our policy by adding varying degrees of noise to the Context-Map, shown in Figure 6. We use: (1) Shift noise, which randomly shifts the map provided to the robot in any direction to simulate localization errors. (2) Cutout noise, which randomly adds patches of free or obstacle space to the map. This simulates giving the robot a outdated map (*i.e.* a map with additional or missing obstacles in the environment).

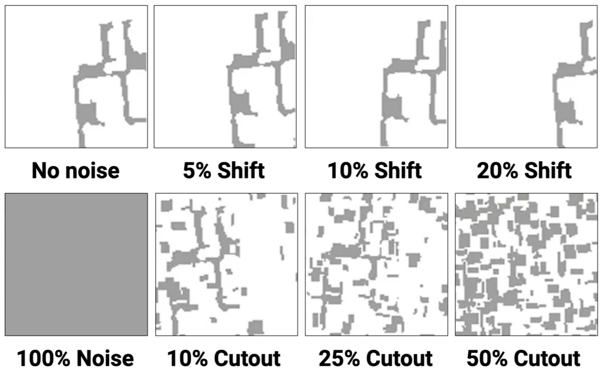

Fig. 6: We degrade the top-down map with shift and cutout noise.

We find that our approach, trained entirely using perfect maps, is surprisingly robust to the noisy maps. At 50% cutout noise, the original top-down map is barely visible (Figure 6), and planning algorithms would be unable to find any feasible path to the goal. In contrast, our policy is still able to navigate to the goal successfully 81.4% of the time (Table III, row 7). When the Context-Map policy is given a map that is completely freespace (100% noise, Table III row 8) the policy behavior regresses to the No-Context policy (79.8 vs. 78.7 SR, and 59.8 vs. 56.4 SPL). Thus, our approach exhibit the best of both worlds – utilizing the information in the map when it's available but never underperforming a map-free approach.

#	Eval Noise		SR ↑	SPL ↑
	Type	Percent		
1	-	-	$95.4_{\pm 0.2}$	$81.0_{\pm 2.4}$
2	Shift	5%	$93.0_{\pm 0.9}$	$75.1_{\pm 2.4}$
3		10%	$84.0_{\pm 0.5}$	$62.5_{\pm 2.3}$
4		20%	$73.2_{\pm 1.6}$	$51.6_{\pm 1.1}$
5	Cutout	10%	$91.5_{\pm 0.2}$	$72.3_{\pm 2.2}$
6		25%	$87.6_{\pm 0.8}$	$67.4_{\pm 2.3}$
7		50%	$81.4_{\pm 2.2}$	$60.0_{\pm 1.0}$
8		100%	$79.8_{\pm 1.2}$	$59.8_{\pm 0.9}$

TABLE III: Context-Map policies trained with perfect maps are surprisingly robust to outdated maps.

IV. Conclusion

In this work, we present a framework for IndoorSim-to-OutdoorReal transfer for navigation. Keys to the system's success are a set of sim-to-real techniques that enable the policy to handle real outdoor environments, as well as the addition of context in the form of a rough sketch provided by a human, which guides the robot's navigation. Our results provide compelling evidence for rejecting the (admittedly reasonable) hypothesis that a new simulator must be designed for every new scenario we wish to study. This is especially important for environments that are challenging to design in simulation, such as the outdoors. This work opens up navigation research to the less explored domain of the rich and diverse outdoors environments.

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abc5986

[2] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abk2822

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[4] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," *CoRL*, 2022.

[5] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, *et al.*, "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" in *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[6] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, *et al.*, "Learning navigation skills for legged robots with learned robot embeddings," in *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[7] N. Yokoyama, S. Ha, and D. Batra, "Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[8] R. Partsey, E. Wijmans, N. Yokoyama, O. Dobosevych, D. Batra, and O. Maksymets, "Is mapping necessary for realistic pointgoal navigation?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 17 232–17 241.

[9] J. Truong, M. Rudolph, N. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation," in *Conference on Robot Learning (CoRL)*, 2022.

[10] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, *et al.*, "Robothor: An open simulation-to-real embodied AI platform," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3164–3174.

[11] M. Deitke, R. Hendrix, L. Weihs, A. Farhadi, K. Ehsani, and A. Kembhavi, "Phone2proc: Bringing robust robots into our chaotic world," *ArXiv*, vol. abs/2212.04819, 2022.

[12] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *International Conference on Learning Representations (ICLR)*, 2020.

[13] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Conference on Robot Learning*. PMLR, 2020, pp. 420–429.

[14] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, *et al.*, "Habitat: A Platform for Embodied AI Research," in *ICCV*, 2019.

[15] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[16] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, *et al.*, "igibson 1.0: a simulation environment for interactive tasks in large realistic scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, p. accepted.

[17] Nvidia, "Isaac Sim," https://developer.nvidia.com/isaac-sim, 2020.

[18] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, *et al.*, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[19] C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, *et al.*, "ThreeDWorld: A platform for interactive multi-modal physical simulation," *arXiv preprint arXiv:2007.04954*, 2020.

[20] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, *et al.*, "SAPIEN: A simulated part-based interactive environment," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[22] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax–a differentiable physics engine for large scale rigid body simulation," *arXiv preprint arXiv:2106.13281*, 2021.

[23] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[24] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, *et al.*, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," *arXiv preprint arXiv:2109.08238*, 2021.

[25] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, *et al.*, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[26] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, *et al.*, "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[27] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, *et al.*, "On Evaluation of Embodied Navigation Agents," *arXiv preprint arXiv:1807.06757*, 2018.

[28] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *CVPR*, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[31] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the cpu," in *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 2018, pp. 16–22.

[32] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.