Gibson Env V2: Embodied Simulation Environments for Interactive Navigation

Fei Xia [1] Chengshu Li [1] Kevin Chen [1] William B. Shen [1] Roberto Martín-Martín [1]
Noriaki Hirose[1] Amir R. Zamir[1,2] Li Fei-Fei[1] Silvio Savarese[1]

[1] Stanford University [2] University of California, Berkeley

http://svl.stanford.edu/gibson2

June 16 2019

1 Introduction

Autonomous navigation is one of the most crucial tasks for mobile agents. The goal is to have the mobile agent reach any location in the environment in a safe and robust manner. Traditionally, robot navigation (including obstacle avoidance) has been addressed with analytical model-based solutions using signals from Lidars or depth sensors [5, 6]. Recently, learning based visual navigation methods have gained popularity because 1) they can perform navigation without accurate localization or metric maps [18, 1], 2) they do not require expensive Lidars or depth sensors [11, 14], 3) they can generalize robustly in previously unseen environments [24, 15].

Despite these benefits, learning-based approaches usually require a large amount of data. Collecting the data through interactions with the real world could be dangerous, costly and time-consuming. A solution to this challenge of learning-based navigation is to learn in simulated environments. In simulation, the agent can collect experiences safely and efficiently: usually one or two orders of magnitude faster than real-time. However, despite recent advances in simulation for robotics [19, 24, 20, 21, 12, 10, 7, 3], it is still less than straightforward to transfer directly what is learned in simulation to the real world. The reason for this is the so-called *sim2real gap*: the (more or less subtle) differences between the simulated and real environments, for example due to the different spatial arrangement of objects or the disparity between real and simulated sensor signals.

Many learning-based approaches rely on simulation for fast policy learning. And different simulation-to-real transfer strategies including photorealistic rendering[16, 4], domain randomization[17] and domain adaptation[2, 23] were proposed. In particular, the Gibson Env[23] is a simulation environment that does photorealistic rendering and additionally provides a pixel-level domain adaptation mechanism to aid with the commonly raised concern of sim-to-real transfer. In Gibson Env the spatial arrangement of objects is realistic because the models of the environments are obtained from the real world. Additionally, the gap between simulated and real visual sensors is bridged through a novel neural network, the Goggles. Thanks to these properties, Gibson Env has demonstrated great sim-to-real transfer performance [9, 13]. However, Gibson Env presents two main limitations that hinder its use for learning-based navigation approaches: 1) the rendering is relatively slow (40-100fps) for large-scale training, which partially defeats the purpose of training in simulation, and 2) the interaction between the agent and the environment is limited only to a planar motion on the floor, while navigation in many real-world scenarios involves more intricate forms of interaction, such as opening doors and pushing away objects that are in the way.

To overcome these limitations, we present **Gibson Env V2**, a significantly improved version over Gibson Env. Gibson Env V2 has significantly faster rendering speed, which makes large-scale training possible. It also allows for more complicated interactions between the agent and

the environment, such as picking up and placing objects, or opening doors and cabinets. This environment opens up new venues for jointly training base and arm policies, allowing researchers to explore the synergy between manipulation and navigation.

This paper provides a brief overview of our updated Gibson Environment. In Section 2, we showcase the new rendering pipeline and dynamic objects handling. In Section 3, we benchmark Gibson Env V2 against Gibson Env V1 and other simulated environments.

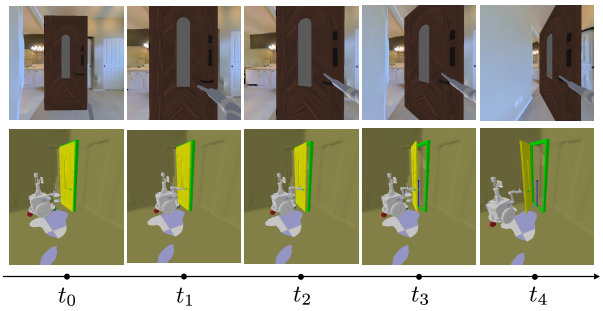

Figure 1: A example trajectory of an agent opening a door with a gripper. **Top row**: Agent's view. **Bottom row**: Physics view.

2 Gibson Env V2

The main bottleneck for rendering speed in Gibson Env V1 is caused by its reliance on image-based rendering (IBR). IBR has the benefit of achieving high photorealism, but it has two main issues. First, in order to render the scene, the system must load images from all available viewpoints and process them on-the-fly. This process is computationally expensive and although modern IBR algorithms can achieve-real time frame rate, on most systems they struggle to render orders of magnitudes faster than real-time[8], which makes learning in a IBR-based environment slow. Secondly, using IBR limits our ability to add interactive objects in the environment because rendering is based on static images.

To address these issues, in Gibson Env V2, we switched to mesh rendering and used a neural network to fix the artifacts and achieve similar photorealism level and faster speed (see Table 1 for rendering speed comparisons). We also enabled addition of interactive objects.

Mesh texturing and rendering Gibson Env V2 requires textured meshes, which were not required for Gibson Env V1. We textured the mesh with globally registered RGB images, using OpenMVS's implementation of MVS-texturing[22]. Because the meshes are already available from fusing depth sensors, we only need to run the texturing step of the traditional MVS-texturing. In Gibson Env V2, we also used view selection, global seam leveling and local seam leveling to achieve high-quality mesh texture. We released all the mesh models as part of the updated software stack.

To keep our rendering pipeline compatible with modern deep learning frameworks, we implemented the entire pipeline in Python with PyOpenGL, PyOpenGL-accelerate, and pybind11 with our custom C++ code. Compared with Gibson Env V1, which did not fully use OpenGL and relied heavily on expensive inter-process communication, we now are able to achieve much lower overhead, resulting in faster rendering speed.

It is worth mentioning that we allow rendered images to directly transfer to a tensor on a GPU in modern deep learning frameworks (such as PyTorch) instead of going through host memory. This effectively reduces device-host memory copies and significantly improves rendering speed (7.9 times frame rate gain at 512×512 resolution and 28.7 times frame rate gain at 1024×1024 resolution).

Interactive Navigation Another main drawback of IBR is that adding objects to existing environments is difficult. Each rendered frame takes pixels from source views, so adding objects would mean manipulating the source images themselves. On the other hand, mesh rendering makes it easy to add novel objects to the scene such as bottles and cans that the agent can interact with. We can also add articulated objects such as doors and cabinets into the environment to allow for more complicated tasks, e.g. interactive navigation or object search.

To simulate physics for the added objects, we used simplified mesh models in Pybullet for collision detection and handling. We synchronized the pose between Pybullet and our own renderer to bypass Pybullet's native rendering. Using our own rendering pipeline gives us improved performance and the freedom to randomize lighting and textures for meshes in the scene. Fig 1 shows a trajectory of an agent interacting with a door. The agent approaches the door, grasps the handle with an end effector and pulls the door open by backing up. To simplify the physics, we

create a universal joint between the end effector and the door handle when they are closer than a certain threshold.

3 Benchmarks

In this section, we benchmarked the rendering speed of Gibson Env V2 against that of Gibson Env V1 and the state-of-the-art 3D simulator Habitat-sim[20]. We also demonstrated the sim-to-real performance on a visual path following task. Note that this section is not an exhaustive benchmark of all aspects of Gibson Env V2, but rather a showcase of some important aspects.

Rendering speed The rendering speed of the improved renderer can be found in Table 1. Due to the reduction of inter-process communication, the rendering speed is much faster than Gibson Env V1, achieving 4x frame rate for RGB and close to 2x frame rate for simultaneous RGB filled, semantic labels and surface normal rendering. For RGB filled rendering, the main bottleneck is running the neural network filler, so the speedup is not as significant. We also compared the rendering speed with Habitat-sim[20]. Since Habitat-sim uses a simplified physics model, we compared it with our rendering-only (no physics) speed for a fair comparison. Our rendering speed is on par with the state-of-the-art simulation engines.

Output	GibsonV2	GibsonV1
Sensor (fps)	1017.4	396.1
RGBD Pre-filled (fps)	264.2	58.5
RGBD Filled (fps)	61.7	30.6
Semantic Only (fps)	279.1	144.2
Surface Normal (fps)	271.1	129.7

Scene	GibsonV2	Habitat-sim
Hillsdale	620.4	752.9
Albertville	422.0	688.2

Table 1: **Rendering speed benchmark**. **Top table:** The table shows rendering speed (frame per second) of our environment under different rendering modes at 256×256 resolution with full physical simulation enabled. **Bottom table:** The table shows the rendering only (no physics) speed of our environment compared with habitat-sim at 640x480 resolution. All benchmarks are run with a single NVIDIA GeForce GTX 1080 Ti Graphics Card

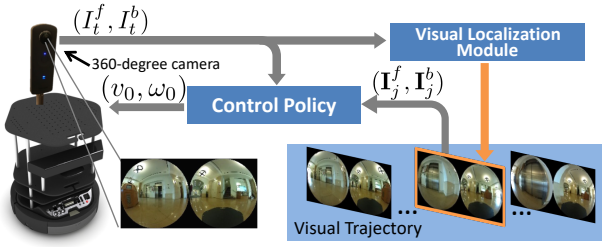

Figure 2: Illustration of sim-to-real transfer experiment on visual path following.

Sim-to-real transfer As a demonstration of the sim-to-real transfer performance of our simulator, we conducted a experiment on a transfer task of visual path following. Fig 2 shows the problem setup: A visual path is defined as a sequence of images collected on a trajectory by the agent, denoted as $(\mathbf{I}_j^f, \mathbf{I}_j^b)$. We use a policy network that takes in the image sequence and the current camera view (I_j^f, I_j^b), and generates velocity commands (v_0, ω_0) for the agent. The images are in pairs because we use a 360 degree field of view camera consisting of two 180 degree field of view images. See [9] for the model and experimental setup details. For the sim-to-real experiment, the visual path is collected in the simulator, and the visual path following task is executed in the real world. There are lighting and object arrangement differences between the simulated environment and the real world. We compared our sim-to-real experiment with a real-to-real experiment in which the visual path is collected in the real world. We evaluated sim-to-real and real-to-real performance in 3 cases, and for each case we repeat the experiment 10 times and calculate success rate and sub-goal coverage rate.

As shown in Table 2, the sim-to-real transfer performance is comparable to the real-to-real performance. Note that the sim-to-real transfer performance is determined by the simulator, the task and the algorithm. There is no guarantee that Gibson Env V2 simulator will work for all algorithms. Rather, this simply shows that, with appropriate algorithmic design, Gibson Env V2 has the potential of transferring to real for navigation tasks.

	Case 1: 6.6 m	Case 2: 8.4 m	Case 3: 12.7 m
Sim2real	0.90 / 0.98	0.80 / 0.87	0.80 / 0.93
Real2real	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00

Table 2: Sim-to-real transfer performance for visual path following task. The table shows success rate (the first number) and sub-goal coverage rate (the second number).

4 Conclusion

In this paper we give a brief overview of an improved version of Gibson Env. It achieved higher rendering speed and is capable of simulating interactive objects in the environment while maintaining the photo-realism of Gibson Env. This environment could accelerate visual navigation policy learning in a realistic indoor environment. More importantly, it opens up possibilities for jointly learning manipulation and navigation policies.

Acknowledgement

We gratefully acknowledge the support of Toyota, Samsung, Nvidia, Stanford Graduate Fellowship and the National Science Foundation. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE1147470. 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.

References

[1] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining optimal control and learning for visual navigation in novel environments. *arXiv preprint arXiv:1903.02531*, 2019. 1

[2] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. IEEE, 2018. 1

[3] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville. Home: A household multimodal environment. *arXiv preprint arXiv:1711.11017*, 2017. 1

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on Robot Learning*, pages 1–16, 2017. 1

[5] F. Flacco, T. Kröger, A. De Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *2012 IEEE International Conference on Robotics and Automation*, pages 338–345. IEEE, 2012. 1

[6] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997. 1

[7] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017. 1

[8] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):231, 2016. 2

[9] N. Hirose, F. Xia, R. Martin-Martin, A. Sadeghian, and S. Savarese. Deep visual mpc-policy learning for navigation. *arXiv preprint arXiv:1903.02749*, 2019. 1, 3

[10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. 1

[11] L. Ke, X. Li, Y. Bisk, A. Holtzman, Z. Gan, J. Liu, J. Gao, Y. Choi, and S. Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. *arXiv preprint arXiv:1903.02547*, 2019. 1

[12] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004. 1

[13] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Neural autonomous navigation with riemannian motion policy. In *International Conference on Robotics and Automation (ICRA)*, 2019. 1

[14] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell. Learning to navigate in cities without a map. *arXiv preprint arXiv:1804.00168*, 2018. 1

[15] D. Mishkin, A. Dosovitskiy, and V. Koltun. Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*, 2019. 1

[16] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016. 1

[17] F. Sadeghi and S. Levine. rl: Real singleimage flight without a single real image. arxiv preprint. *arXiv preprint arXiv:1611.04201*, 12, 2016. 1

[18] H. Salman, P. Singhal, T. Shankar, P. Yin, A. Salman, W. Paivine, G. Sartoretti, M. Travers, and H. Choset. Learning to sequence robot behaviors for visual navigation. *arXiv preprint arXiv:1803.01446*, 2018. 1

[19] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017. 1

[20] M. Savva, A. Kadian, O. Maksymets, et al. Habitat: A platform for embodied ai research. *arXiv preprint arXiv*, 2019. 1, 3

[21] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

[22] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! — Large-scale texturing of 3D reconstructions. In *Proceedings of the European Conference on Computer Vision*. Springer, 2014. 2

[23] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018. 1

[24] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE, 2017. 1