Comparing and Mitigating Scene Reconstruction Errors in the Matterport3D Dataset

Jacob Krantz **Damon George**

Abstract

AI agents that can interact with physical environments are often trained in perceptually rich simulations before being deployed into the real world. To mimic affordances of physical environments such as unconstrained movement and vision, simulated 3D scenes can be reconstructed from real-world scenes using sparsely-captured high-resolution RGBD images. In this work, we analyze the discrepancy of visual fidelity between camera captures in reconstructed scenes and raw RGBD images of the Matterport3D dataset, home to many embodied AI tasks such as vision-and-language navigation, PointGoal navigation, and embodied question answering. We show that a classifier can be trained to discriminate between reconstructed scene viewpoints and raw RGBD images. Further, we rank scenes by reconstruction quality using a keypoint matching algorithm. Finally, we apply a domain adaptation method, "*Goggles*", to improve the visual quality of agent perception used previously on the Gibson dataset.

1 Dataset Description

To perform our analysis, we consider the Matterport3D dataset[1], which contains 10,800 image viewpoints covering 90 different real-world scenes with constructed 3D meshes for each scene (3). At each viewpoint, a sequence of 12 images were captured, each at a heading rotated 30 degrees away from the previous. Together, these 12 images account for a complete 360 degree rotation about the vertical axis of a viewpoint. This gives us $10800 \times 12 = 129,600$ original images to work with. The original collection process was also repeated for a camera angled slightly up and for a camera angled slightly down (so 36 images per viewpoint), but for our project we consider just the flat elevation.

Custom Dataset Curation. Given the original 129,600 Matterport 3D images, we sought to collect a matching image for each from the 3D mesh reconstruction. We loaded the meshes into the Habitat Simulator (13), computed a mapping of camera parameters, and collected matching images. An example of a matched viewpoint in Matterport3D and in Habitat is shown in Fig. 1. To the best of our knowledge, this viewpoint mapping has not been previously performed. The most related work mapped just the coordinates of the camera, ignoring heading and intrinsics (10).

Mapping from Matterport3D to Habitat. For each viewpoint, we needed to transfer the coordinate location and the heading. The Matterport3D dataset provides a 4x4 extrinsic pose matrix which we were able to extract an (X,Y) translation from. Given an (X,Y,Z) coordinate with Z up, we then rotated the position to have Y up which is the definition in Habitat. To map the camera heading, we took a heading scalar from Matterport3D and created a quaternion for Habitat. For camera intrinsics, we simply copied the height and width resolutions as well as VFOV and HFOV settings (H: 640, W: 480, VFOV: 60).

Collecting Matching Images. For each viewpoint successfully transferred to Habitat, we collected 12 images about the vertical axis, ensuring that the heading of each image from Habitat matched as

[1] `https://niessner.github.io/Matterport/`

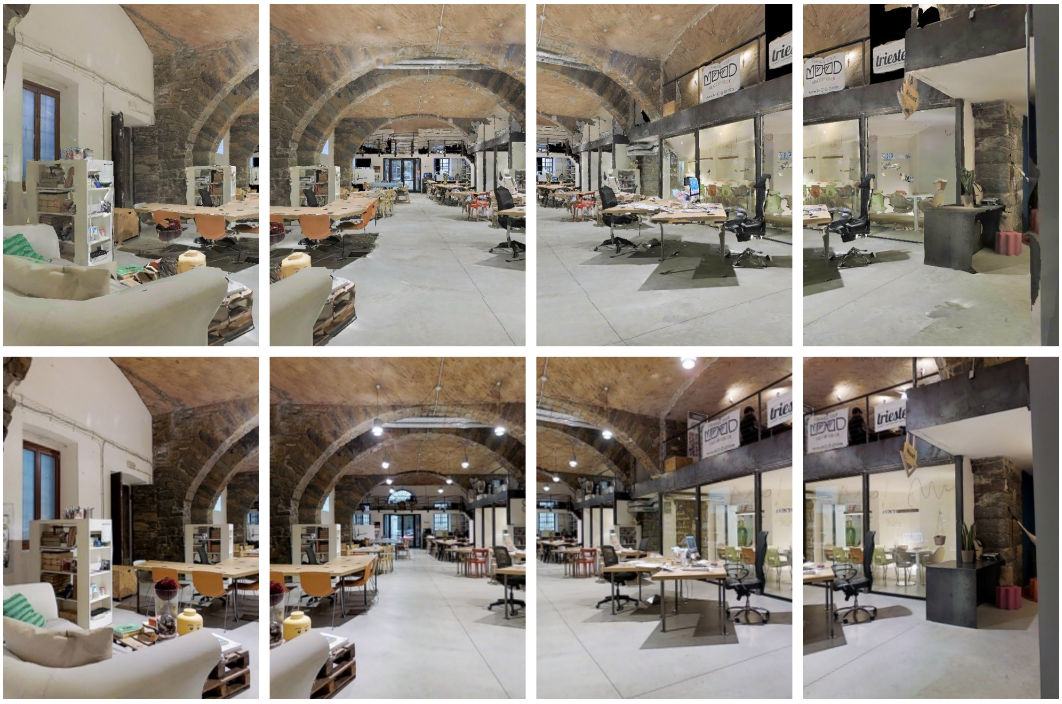


Figure 1: Example images from our dataset. The top images are from a reconstructed viewpoint captured in the Habitat Simulator and the bottom are from Matterport3D. Notice differences such as the presence of light fixtures in the Matterport3D images and the jagged lines in the reconstructions.

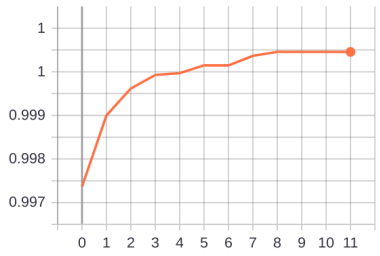
closely as possible to the corresponding heading in Matterport. Altogether, this process resulted in 10,362 matched viewpoints (96% mapping rate, 124,344 matched images) covering all 90 scenes of the Matterport3D dataset.

Dataset Splits. For all tasks we consider that require optimizing parameters (e.g. classification), we need splits of the dataset. We follow the splits from the original Matterport3D paper (3) which has been subsequently adopted by derivative datasets (1). To this end, we split the dataset into train (61 environments, 179,232 images), unseen validation (11 environments, 22,656 images), and test (18 environments, 46,800 images).

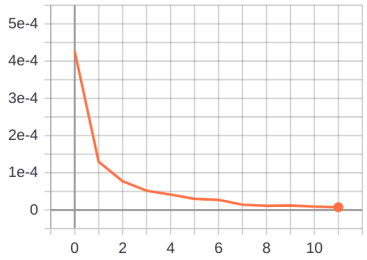
Differences From Our First Proposal. In our project proposal, we proposed collecting panoramic images from each of the viewpoints. Upon looking into the Matterport3D dataset, we found that these panoramic images were stitched from images captured by rotation about an axis as described above. In an effort to work with more raw collected photos (which is common in projects using this dataset (5)), we chose to consider the set of regular images instead of stitched panoramas.

2 Tasks

Our synthesized Matterport3D pairs dataset allows us to study the discrepancies between raw images and images taken from within a 3D mesh reconstruction. This study is motivated by the application of training embodied AI agents within photo-realistic 3D environments. We eventually want these agents to perform in the real world, however training them in the real world is costly and impractical given current learning algorithms. Developing these agents in simulated environments allows for drastically decreased training time and avoids the complications of physical robotic hardware. However, existing simulated 3D environments are not perceptually identical to real world environments. Despite being reconstructed from real-world camera captures, these simulated environments, such as the meshes in Matterport3D, contain holes of missing texture from camera occlusions, lack of visual fidelity as seen in light fixtures and wall decorations caused by combining multiple camera angles, and differences in lighting and color.



(a) Accuracy vs. epochs for val unseen.



(b) Loss vs. epochs for val unseen.

Figure 2: Training and evaluation curves for our reconstruction discrimination task. We find that our network quickly reaches a high validation accuracy (first epoch), and learns to correctly classify nearly all outliers after 8 epochs.

Such errors are reasons for the existence of a domain shift problem; agents that are trained in the domain of simulated environments may not be well suited to immediate transfer to the domain of reality. As such, many works that consider transfer from simulation to reality (Sim2Real) leverage domain adaptation methods such as domain randomization to limit the drop in performance (15; 2). Recent work has created a Sim2Real Correlation Coefficient (SRCC) to measure the predictability of simulation performance in reality (8).

In the tasks we perform on our paired dataset, our contributions are threefold:

- We confirm that a domain gap exists between raw Matterport 3D images and images captured in scene reconstructions.
- We rank each of the 90 scenes by similarity to original camera captures according to a keypoint matching method.
- We implement a perceptual improvement model based on the "*Goggles*" method proposed in *Xia et al., 2018* (17) and show that it alters mesh images to appear more like raw images.

2.1 Reconstruction Discrimination

We first seek to determine if there is a perceptual difference between reconstructed images and raw images that can be learned by a classifier. The intuitive answer is yes, such a difference should be easily learnable because the issues outlined above and by observing examples like Fig. 1. We note that some of the reconstructed images are challenging to discriminate from the human perspective. Training a classifier to perform this task will give a more nuanced indication as to how large the gap is between reconstructed and real.

Method. We can frame this task as a binary classification problem by flattening our dataset of 124,344 image pairs to 248,688 images each labeled either raw or reconstructed. We maintain the same scene-based dataset splits. We note that our dataset is class-balanced by nature of our collection process. To solve this classification problem, we implement a convolutional neural network (ResNet-50 (6)) and initialized its parameters with pre-trained weights from ImageNet (4). We replace the final 1000-dimensional class layer from the standard ResNet-50 with a single value activated by the sigmoid function. The evaluation metric for this task is average classification accuracy.

We choose the Adam optimizer (9) to minimize a binary cross-entropy loss function using our $\sim$23M network parameters. For hyperparameters, we select a small learning rate (2.5e-7) so as not to lose useful pre-trained network connections. We use the largest batch size that will fit on our GPU (40), and set the epoch limit to 10. After each training epoch, we evaluate the network on the validation unseen split. We implemented our network and training regime with PyTorch (12) and used TorchVision for the ResNet definition and parameter loading. We ran our experiments on a NVIDIA Tesla V100 GPU where each epoch through our training set took 1.5 hours and evaluation of the unseen validation and test sets took 5 minutes each.

Results. As seen in Fig. 2, our network converged to 100.00% accuracy on the validation set after 8 epochs so we stopped training. We then evaluated our network on the test split, garnering a 99.99% accuracy. There may be some sort of pixel regularities that make this classification an easier task than

Table 1: Summary of the scene ranking result in which all 90 scenes are ranked according to their keypoint matching image retrieval accuracy. The three best and worst scenes are shown.

Scene	Accuracy (%)
8WUmhLawc2A	32.89
JeFG25nYj2p	26.79
EDJbREhghzL	25.36

(a) Scenes with the highest image retrieval accuracy.

Scene	Accuracy (%)
D7G3Y4RVNrH	9.38
82sE5b5pLXE	8.66
Uxmj2M2itWa	5.95

(b) Scenes with the lowest image retrieval accuracy.

we had expected, such as abrupt edges in the reconstructions. Seeing that the accuracy of our network is nearly perfect, we find that not only do some of the reconstructed images have imperfections, but all of them do. This motivates effort into reconstruction perception improvement. Confirming the extent of this discrepancy allows us to move into more interesting research questions such as 1) which scenes have better reconstructions (our keypoint method) and 2) how can we mitigate the perceptual discrepancies (reconstruction improvement). Also, we note that since we have created the dataset ourselves, there are no existing baselines to compare our discrimination results to.

2.2 Scene Ranking

We also aim to rank the 90 scenes based on the similarity of their reconstructions to their corresponding raw images. A use case for this would be to train embodied agents in just the top X% of scene reconstructions to avoid the largest domain gaps. We can formulate this task as a per-scene image retrieval problem using the reconstructed images as the queries and the raw images from the corresponding scene as the matching set. Although this may not be the most natural way to frame this problem, it does allow us to incorporate keypoint detection and matching (two important topics from class) into this project.

Method. Formulating the scene ranking as a keypoint matching problem first entails detecting keypoints in all images and computing their deep features. We use the OpenCV implementation of SIFT (11) for the keypoint detection, and we implement a simple PyTorch CNN from class homework as the deep feature generator. Using a PyTorch Dataloader to parallelize the detection task across 8 processes, generating deep features of 100 keypoints for all 248,688 images takes about 5 hours: under 4 minutes per scene. After generating deep features for all keypoints, we split the dataset by scene. There are 90 scenes, with each containing on average around 1380 image pairs. For each reconstructed image in a given scene, we perform one-to-one matching using the Hungarian algorithm with all the raw images from the same scene. We express this matching problem as the following: Given reconstructed query image q and raw image r, we compute the k-th deep feature from q as f_k^q and the l-th deep feature from r as f_l^r. We then compute the cost matrix C containing the cost $c_{k,l}$ of all feature pairs (f_k^q, f_l^r). The cost is defined as

$$c_{k,l} = \frac{1}{2}\left(1 - \frac{(f_k^q)^\top f_l^r}{\|f_k^q\|_2 \, \|f_l^r\|_2}\right). \tag{1}$$

We then use the Hungarian algorithm to find the binarized values $X^* = [x_{k,l}^*]$ that minimize $trace(C^\top X)$, such that $\forall l, \sum_k x_{k,l} = 1$ and $\forall k, \sum_l x_{k,l} = 1$. To perform this matching, we use the SciPy implementation of the Hungarian algorithm. After finding X^*, we compute the total cost of the matching for each pair of images in the scene as $d(q,r) = \sum_{k,l} c_{k,l} x_{k,l}^*$. The raw image with the minimum total matching cost is returned for each reconstructed query image, and since each reconstructed image is only paired with a single raw image, we can easily compute the retrieval accuracy for the entire scene. This accuracy is then used to rank all 90 scenes. After parallelizing the matching across multiple processes as well, the implementation takes about 20 minutes per scene, resulting in a total running time of 36 hours for this entire task.

Results. In Tab. 1, we summarize the scene ranking result by showing the image retrieval accuracy of the three best and worst scenes. As shown, the highest accuracy achieved is about 33% while the worst accuracy is 6%. The average accuracy across all 90 scenes was 19.31%, which is clearly a poor result. The specific scenes are not currently important to our analysis, but this result does

Table 2: Comparison of stages of reconstruction improvement to the raw image under metrics mean squared error (MSE) and structural similarity (SSIM). Each measurement is ± standard error. We find that despite a higher MSE, F(Representation) (function f) has a better SSIM than Reconstruction.

	val_unseen		test	
	MSE↓	SSIM↑	MSE↓	SSIM↑
Reconstruction	**14.361**±**0.032**	0.5186±0.0014	**14.410**±**0.027**	0.4958±0.0011
Autoencoded Reconstruction	14.453±0.032	0.4948±0.0013	14.463±0.026	0.4664±0.0010
F(Reconstruction)	15.331±0.030	**0.5870**±**0.0011**	15.451±0.022	**0.5746**±**0.0009**

highlight the discrepancy between the reconstructed and raw images. Since keypoints describe the essential and unique characteristics of an image, this low image retrieval accuracy shows that the mesh reconstructions have lost, distorted, and/or suppressed the important characteristics of the original raw images. This scene ranking result clearly illustrates the need for improvement in the reconstructions and may also help future work in focusing improvement efforts on scenes with the worst image retrieval accuracies.

2.3 Perception Quality Improvement

Discrepancies between a scene reconstruction and original images can be considered to be a domain gap and mitigated by domain adaptation. One approach to mitigating this domain gap is to formulate the domains created by the simulated images S and raw images W as a joint space and minimizing the space between them (14). We originally proposed to follow the "*Goggles*" solution proposed by Xia et al. (17), which is to learn two functions: f (which maps simulated image s to raw image w) and u (which maps w to $u(s)$). However, due to the time and compute resources it took to train the initial function f, we were unable to train function u. What follows is the experiment details and results for training just the function f.

Method. In order to train function f, the original paper first trained it as an autoencoder of simulated image s. Thus, the initial training objective sought to minimize the mean squared error between s and $f(s)$ as such: $J = \text{MSE}(f(s), s)$. After we trained this to convergence, we changed the objective function to optimize f to output an image more similar to the real image w:

$$J = \text{Perception}(f(s), w) + \lambda \text{MSE}(f(s), w).$$

The Perception loss term aims to minimize the distance between the embeddings of $f(s)$ and w using extracted VGG16 features at various levels of feature abstraction. For all experiments we set $\lambda = 0.05$, which is the value used in the original paper. We train to convergence on this new objective, which is much harder than the initial objective. Critically, the reconstructions are missing information that would otherwise be necessary to recreate the orignal images. As such, a gap necessarily exists for unseen images even after $f(s)$ reaches convergence. This is where mapping the original images to a joint space is necessary (function u). We expect that if we had time to train function u, our results would be stronger as theoretically the distance between $f(s)$ and $u(w)$ can reach zero.

Our implementation details are as follows. We use the exact architecture for f as Xia et al.,(17) and apply the same loss functions (with the same pretrained VGG16 weights). For these steps, we borrow code from their implementation [2]. We set the learning rate to 2×10^{-4} for all training settings and use the Adam optimizer. We do not train the perceptual VGG loss function. We train on the largest batch size we can fit on our GPU which is 30. Training f as an autoencoder of image s took 24 hours for 6 epochs. Training f toward image w took 48 hours for 10 epochs. We had to spread this training across two GPUs (NVIDIA Tesla V100) to fit the VGG inference.

Metrics. To compare the quality of our various image sets to the raw images, we compute average SSIM and MSE scores (16). SSIM, or structural similarity index, is an estimate of perceived similarity between an image and its reference image that considers texture regions, where a higher score is better. MSE, or mean squared error, just computes a mean error over each pixel channel value where lower is better. We do not include peak signal to noise ratio (PSNR) because SSIM is a good predictor of it(7). We use the SSIM implementation from the Scikit-Image library [3].

[2] `https://github.com/StanfordVL/GibsonEnv`

[3] `https://scikit-image.org/`

Figure 3: A qualitative example of each stage of our reconstruction improvement from val_unseen. Left: reconstruction. Left Center: f as an autoencoder. Right Center: f. Right: original image.

Results. In Tab. 2, we see that although function f increases the MSE of the original reconstructed images, it more importantly increases the SSIM score. Knowing that SSIM is a metric of perceptual similarity, this shows that we were indeed able to improve the perception quality of the reconstructed images. Notice that in Fig. 3 the function f produces a blurry-looking output. However upon closer inspection, it has learned a valuable skill in filling in mesh holes. The most prominent case in this example is to the right of the central pillar where the wall color is correctly filled in. Further, the reconstruction has disrupted lines in the ceiling tiles which are filled in and straightened by f. The blur in the output image clearly shows a loss of information and detail, which results in the heightened MSE score; however, the important structures and textures in the image are no longer distorted, which explains the increase in SSIM score.

Discussion. After performing this Goggles experiment, we learned a few takeaways. First, the mapping from original images to reconstructed ones, while of good quality for most images, was slightly off for some due to reconstruction errors in the mesh. This can account for situations where we observe two lines next to each other (aka "double vision") instead of just the one line that occurs in slightly different places in the reconstructed image vs the original ones. Our experiments also differ from the that of the original Goggles paper in that we applied the method to 480x640x3 dimensional images instead of 256x256x3, which in hindsight is a harder task. If we were to repeat this experiment, we would use the smaller image size because 1) we can use a larger batch size, 2) less visual fidelity makes the task easier, and 3) most embodied tasks use a 256x256 or smaller image (10; 1).

3 Conclusion

In this work, we attempted to analyze and improve scene reconstruction errors in the Matterport3D dataset. To do this, we first created a dataset consisting of raw and reconstructed image pairs by mapping viewpoints from the original dataset to the mesh reconstructions in the Habitat Simulator. Next, we confirmed the domain gap between the raw and reconstructed images by training a ResNet-50 to perform binary classification on all images, which was able to achieve 99.99% test accuracy after only 8 epochs. We also analyzed this domain gap by ranking all 90 scenes in the dataset according to their keypoint matching image retrieval accuracy. The poor average matching accuracy further illustrated the discrepancies between the reconstructions and raw images. And lastly, we implemented half of the Goggles solution in which we learned a mapping from simulated images to raw images. This did improve the structural similarity of the reconstructions, but time and resource constraints prevented us from using this algorithm to its fullest potential.

Our team of two for this project was Jacob Krantz and Damon George. The distribution of labor was: Damon collected the Matterport3D images, Jacob collected the reconstructed Habitat images and performed the ResNet reconstruction discrimination task, Damon performed the scene ranking analysis, Jacob did the perception quality improvement experiment, and both contributed to writing the report.

References

[1] Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., van den Hengel, A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3674–3683 (2018)

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

[3] Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3D: Learning from RGB-D data in indoor environments. In: 3DV (2017), MatterPort3D dataset license available at: `http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf`

[4] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)

[5] Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., Darrell, T.: Speaker-follower models for vision-and-language navigation. In: Advances in Neural Information Processing Systems. pp. 3314–3325 (2018)

[6] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[7] Hore, A., Ziou, D.: Image quality metrics: Psnr vs. ssim. In: 2010 20th International Conference on Pattern Recognition. pp. 2366–2369. IEEE (2010)

[8] Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., Batra, D.: Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation. In: arXiv (2019)

[9] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[10] Krantz, J., Wijmans, E., Majumdar, A., Batra, D., Lee, S.: Beyond the nav-graph: Vision-and-language navigation in continuous environments. arXiv preprint arXiv:2004.02857 (2020)

[11] Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**, 91–110 (11 2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94

[12] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)

[13] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al.: Habitat: A platform for embodied ai research. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9339–9347 (2019)

[14] Sener, O., Song, H.O., Saxena, A., Savarese, S.: Learning transferrable representations for unsupervised domain adaptation. In: Advances in Neural Information Processing Systems. pp. 2110–2118 (2016)

[15] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 23–30. IEEE (2017)

[16] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)

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