Gibson Env: Real-World Perception for Embodied Agents

Fei Xia$^{*}_{1}$ Amir R. Zamir$^{*}_{1,2}$ Zhiyang He$^{*}_{1}$ Alexander Sax$_1$ Jitendra Malik$_2$ Silvio Savarese$_1$

^{1} Stanford University ^{2} University of California, Berkeley

http://gibson.vision/

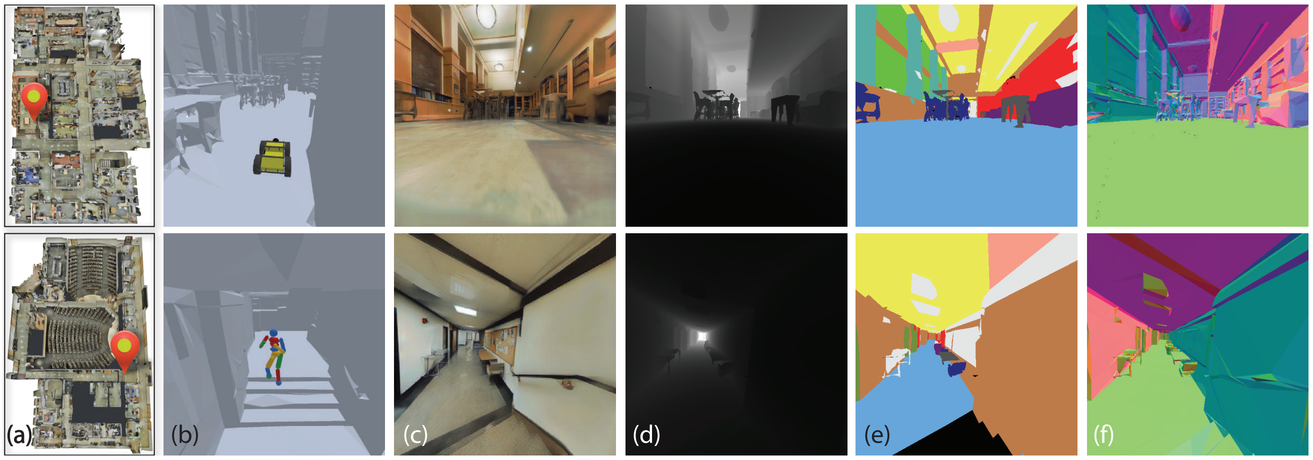

Figure 1: Two agents in *Gibson Environment* for real-world perception. The agent is active, embodied, and subject to constraints of physics and space (a,b). It receives a constant stream of visual observations as if it had an on-board camera (c). It can also receive additional modalities, e.g. depth, semantic labels, or normals (d,e,f). The visual observations are from real-world rather than an artificially designed space.

Abstract

*Developing visual perception models for active agents and sensorimotor control in the physical world are cumbersome as existing algorithms are too slow to efficiently learn in real-time and robots are fragile and costly. This has given rise to learning-in-simulation which consequently casts a question on whether the results transfer to real-world. In this paper, we investigate developing **real-world perception for active agents**, propose **Gibson Environment** [1] for this purpose, and showcase a set of perceptual tasks learned therein. Gibson is based upon virtualizing real spaces, rather than artificially designed ones, and currently includes over 1400 floor spaces from 572 full buildings. The main characteristics of Gibson are: I. being from the real-world and reflecting its semantic complexity, II. having an internal synthesis mechanism "Goggles" enabling deploying the trained models in real-world without needing domain adaptation, III. embodiment of agents and making them subject to constraints of physics and space.*

[1]Named after **JJ Gibson**, the author of *Ecological Approach to Visual Perception*, 1979. "We must perceive in order to move, but we must also move in order to perceive" – JJ Gibson [36]

*Authors contributed equally.

1. Introduction

We would like our robotic agents to have compound perceptual and physical capabilities: a drone that autonomously surveys buildings, a robot that rapidly finds victims in a disaster area, or one that safely delivers our packages, just to name a few. Apart from the application perspective, the findings supportive of a close relationship between visual perception and motion are prevalent on various fronts: evolutionary and computational biologists have hypothesized a key role for intermixing perception and locomotion in development of complex behaviors and species [61, 91, 22]; neuroscientists have extensively argued for a hand in hand relationship between developing perception and being active [83, 42]; pioneer roboticists have similarly advocated entanglement of the two [14, 15]. This all calls for developing principled perception models specifically with active agents in mind.

By perceptual active agent, we are generally referring to an agent that receives a visual observation from the environment and accordingly effectuates a set of actions which can lead a physical change in the environment (∼manipulation) and/or the agent's own particulars (∼locomotion). Developing such perceptual agents entails the questions of *how* and *where* to do so.

On the *how* front, the problem has been the focus of a broad set of topics for decades, ranging from classical control [64, 12, 50] to more recently sensorimotor control [33, 54, 55, 5], reinforcement learning [6, 73, 74], acting by prediction [28], imitation learning [23], and other concepts [59, 100]. These methods generally assume a sensory observation from the environment is given and subsequently devise one or a series of actions to perform a task.

A key question is *where* this sensory observation should come from. Conventional computer vision datasets [32, 26, 57] are passive and static, and consequently, lacking for this purpose. Learning in the physical world, though not impossible [38, 7, 55, 63], is not the ideal scenario. It would bound the learning speed to real-time, incur substantial logistical cost if massively parallelized, and discount rare yet important occurrences. Robots are also often costly and fragile. This has led to popularity of learning-in-simulation with a fruitful history going back to decades ago [64] and remaining an active topic today. The primary questions around this option are naturally around *generalization from simulation to real-world*: how to ensure I. the *semantic complexity* of the simulated environment is a good enough replica of the intricate real-world, and II. the *rendered visual observation* in simulation is close enough to what a camera in real-world would capture (photorealism).

We attempt to address some of these concerns and propose Gibson, a virtual environment for training and testing *real-world* perceptual agents. An arbitrary agent, e.g. a humanoid or a car (see Fig. 1) can be imported, it will be then embodied (i.e. contained by its physical body) and placed in a large and diverse set of real spaces. The agent is subject to constraints of space and physics (e.g. collision, gravity) through integration with a physics engine, but can freely perform any mobility task as long as the constraints are satisfied. Gibson provides a stream of visual observation from arbitrary viewpoints as if the agent had an on-board camera. Our novel rendering engine operates notably faster than real-time and works given sparsely scanned spaces, e.g. 1 panorama per 5-10 m^2.

The main goal of Gibson is to facilitate transferring the models trained therein to real-world, i.e. holding up the results when the stream of images switches to come from a real camera rather than Gibson's rendering engine. This is done by: first, resorting to the world itself to represent its own semantic complexity [81, 14] and forming the environment based off of scanned real spaces, rather than artificial ones [84, 48, 46]. Second, embedding a mechanism to dissolve differences between Gibson's renderings and what a real camera would produce. As a result, an image coming from a real camera vs the corresponding one from Gibson's rendering engine look statistically indistinguishable to the agent, and hence, closing the (perceptual) gap. This is done by employing a neural network based rendering approach which jointly trains a network for making renderings look more like real images (forward function) as well as a network which makes real images look like renderings (inverse function). The two functions are trained to produce equal outputs. The inverse function resembles deployment-time *corrective glasses* for the agent, thus we call it *Goggles*.

Finally, we showcase a set of active perceptual tasks (local planning for obstacle avoidance, distant navigation, visual stair climbing) learned in Gibson. Our focus in this paper is on the vision aspect only. The statements should not be viewed to be necessarily generalizable to other aspects of learning in virtual environments, e.g. physics simulation.

Gibson Environment and our software stack are available to public for research purposes at http://gibson.vision/.

2. Related Work

Active Agents and Control: As discussed in Sec.1, operating and controlling active agents have been the focus of a massive body of work. A large portion of them are non-learning based [50, 27, 49], while recent methods have attempted learning visuomotor policies end-to-end [100, 54] taking advantage of imitation learning [69], reinforcement learning [74, 41, 73, 41, 5, 6], acting by prediction [28] or self-supervision [38, 63, 28, 62, 43]. These methods are all potential users of (ours and other) virtual environments.

Virtual Environments for Learning: Conventionally vision is learned in static datasets [32, 26, 57] which are of limited use when it comes to active agent. Similarly, video datasets [53, 66, 95] are pre-recorded and still passive. Virtual environments have been a remedy for this, classically [64] and today [100, 34, 29, 79, 44, 10, 8, 68, 92]. Computer games, e.g. Minecraft [46], Doom [48] and GTA5 [65] have been adapted for training and benchmarking learning algorithms. While these simulators are deemed reasonably effective for certain planning or control tasks, the majority of them are of limited use for perception and suffer from over simplification of the visual world due to using synthetic underlying databases and/or rendering pipeline deficiencies. Gibson addresses some of these concerns by striving to target perception in real-world via using real spaces as its base, a custom neural network based view synthesizer, and a baked-in adaption mechanism, *Goggles*.

Domain Adaptation and Transferring to Real-World: With popularity of simulators, different approaches for domain adaption for transferring the results to real world has been investigated [11, 25, 85, 71, 89, 93], e.g. via domain randomization [70, 89] or forming joint spaces [77]. Our approach is relatively simple and makes use of the fact that, in our case, large amounts of paired data for target-source domains are available enabling us to train forward and inverse models to form a joint space. This makes us a baked-in mechanism in our environment for adaption, minimizing the need for additional and custom adaptation.

View Synthesis and Image-Based Rendering: Rendering novel views of objects and scenes is one of the classic

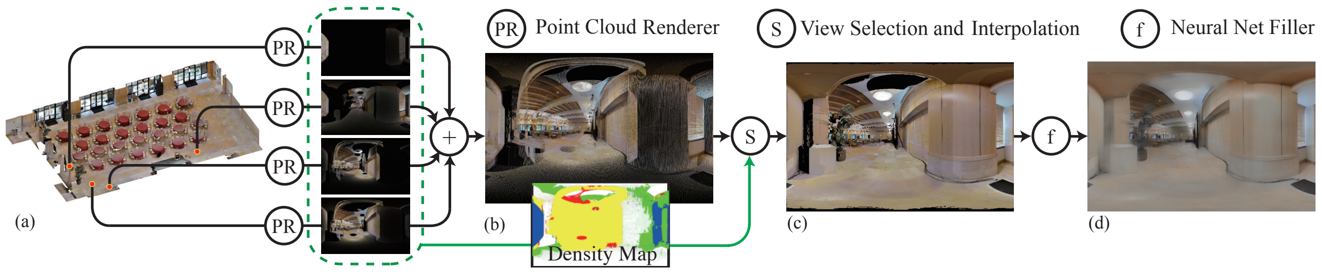

Figure 2: Overview of our view synthesis pipeline. The input is a sparse set of RGB-D Panoramas with their global camera pose. (a,b) Each RGB-D panorama is projected to the target camera pose and rendered. (b) View Selection determines from which panorama each target pixel should be picked, favoring panoramas that provide denser pixels for each region. (c) The pixels are selected and local gaps are interpolated with bilinear sampling. (d) A neural network (f) takes in the interpolated image and fills in the dis-occluded regions and fixes artifacts.

problems in vision and graphics [76, 80, 87, 21, 56]. A number of relevantly recent methods have employed neural networks in a rendering pipeline, e.g. via an encoder-decoder like architecture that directly renders pixels [30, 52, 88] or predicts a flow map for pixels [99]. When some from of 3D information, e.g. depth, is available in the input [39, 58, 18, 78], the pipeline can make use of geometric approaches to be more robust to large viewpoint changes and implausible deformations. Further, when multiple images in the input are available, a smart selection mechanism (often referred to as Image Based Rendering) can help with lighting inconsistencies and handling more difficult and non lambertian surfaces [40, 60, 90], compared to rendering from a textured mesh or as such entirely geometric methods. Our approach is a combination of above in which we geometrically render a base image for the target view, but resort to a neural network to correct artifacts and fill in the dis-occluded areas, along with jointly training an inverse function for mapping real images onto the synthesized one.

3. Real-World Perceptual Environment

Gibson includes a neural network based view synthesis (described in Sec. 3.2) and a physics engine (described in Sec. 3.3). The underlying scene database and integrated agents are explained in sections 3.1 and 3.3, respectively.

3.1. Gibson Database of Spaces

Gibson's underlying database of spaces includes 572 full buildings composed of 1447 floors covering a total area of 211k m^2. Each space has a set of RGB panoramas with global camera poses and reconstructed 3D meshes. The base format of the data is similar to 2D-3D-Semantics dataset [9], but is more diverse and includes 2 orders of magnitude more spaces. This dataset is released as asset files within Gibson[2].

We have also integrated 2D-3D-Semantics dataset [9] and Matterport3D [16] in Gibson for optional use.

[2]Stanford AI lab has the copyright to all models.

3.2. View Synthesis

Our view synthesis module takes a sparse set of RGB-D panoramas in the input and renders a panorama from an arbitrary novel viewpoint. A 'view' is a 6D camera pose of x, y, z Cartesian coordinates and roll, pitch, yaw angles, denoted as θ, ϕ, γ. An overview of our view synthesis pipeline can be seen in Fig. 2. It is composed of a geometric point cloud rendering followed by a neural network to fix artifacts and fill in the dis-occluded areas, jointly trained with an inverse function. Each step is described below:

Geometric Point Cloud Rendering. Scans of real spaces include sparsely captured images, leading to a sparse set of sampled lightings from the scene. The quality of sensory depth and 3D meshes are also limited by 3D reconstruction algorithms or scanning devices. Reflective surfaces or small objects are often poorly reconstructed or entirely missing. All these prevent simply rendering from textured meshes to be a sufficient approach to view synthesis.

We instead adopt a two-stage approach, with the first stage being geometrically rendering point clouds: the given RGB-D panoramas are transformed into point clouds and each pixel is projected from equirectangular coordinates to Cartesian coordinates. For the desired target view $v_j = (x_j, y_j, z_j, \theta_j, \phi_j, \gamma_j)$, we choose the nearest k views in the scene database, denoted as $v_{j,1}, v_{j,2}, \ldots, v_{j,k}$. For each view $v_{j,i}$, we transform the point cloud from $v_{j,i}$ coordinate to v_j coordinate with a rigid body transformation and project the point cloud onto an equirectangular image. The pixels may open up and show a gap in-between, when rendered from the target view. Hence, the pixels that are supposed to be occluded may become visible through the gaps. To filter them out, we render an equirectangular depth as seen from the target view v_j since we have the full reconstruction of the space. We then do a depth test and filter out the pixels with a difference $> 0.1m$ in their depth from the corresponding point in the target equirectangular depth. We now have sparse RGB points projected in equirectangulars for each reference panorama (see Fig. 2 (a)).

The points from all reference panoramas are aggregated to make one panorama using a locally weighted mixture (see Density Map in Fig. 2 (b)). We calculate the point density for each spatial position (average num-

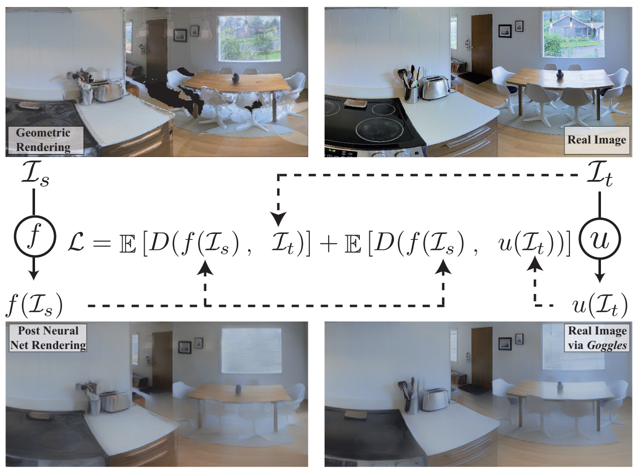
Geometric Rendering
Real Image
$\mathcal{I}_s$
$\mathcal{I}_t$
f
$\mathcal{L} = \mathbb{E}\left[D(f(\mathcal{I}_s), \mathcal{I}_t)\right] + \mathbb{E}\left[D(f(\mathcal{I}_s), u(\mathcal{I}_t))\right]$
u
$f(\mathcal{I}_s)$
$u(\mathcal{I}_t)$
Post Neural Net Rendering
Real Image via *Goggles*

Figure 3: Loss configuration for neural network based view synthesis. The loss contains two terms. The first is to transform the renderings to ground truth target images. The second is to alter ground truth target images to match the transformed rendering. A sample case is shown.

ber of points per pixel) of each panorama, denoted as $d_1, \ldots, d_k$. For each position, the weight for view i is $\exp(\lambda_d d_i)/\sum_m \exp(\lambda_d d_m)$, where λ_d is a hyperparameter. Hence, the points in the aggregated panorama are adaptively selected from all views, rather than superimposed blindly which would expose lighting inconsistency and misalignment artifacts.

Finally, we do a bilinear interpolation on the aggregated points in one equirectangular to reduce the empty space between rendered pixels (see Fig. 2 (c)).

See the first row of Fig. 6 which shows the so-far output still includes major artifacts, including stitching marks, deformed objects, or large dis-occluded regions.

Neural Network Based Rendering. We use a neural network, f, to fix artifacts and generate a more real looking image given the output of the geometric point cloud rendering. We employ a set of key novelties to produce good results efficiently, including a stochastic identify initialization and adding color moment matching in perceptual loss.

Architecture: The architecture and hyperparameters of our convolutional neural network f are detailed in the supplementary material. We utilize dilated convolutions [96] to aggregate contextual information. We use a 18-layer network, with 3×3 kernels for dilated convolution layers. The maximal dilation is 32. This allows us to achieve a large receptive field but not shrink the size of the feature map by too much. The minimal feature map size is $\frac{1}{4} \times \frac{1}{4}$ of the original image size. We also use two architectures with the number of kernels being 96 or 512, depending on whether speed or quality is prioritized.

Identity Initialization: Though the output of the point cloud rendering suffers from notable artifacts, it is yet quite close to the ground truth target image numerically. Thus, an identity function (i.e. input image=ouput image) is a good place for initializing the neural network f at. We develop a stochastic approach to initializing the network at identity, to keep the weights nearly randomly distributed. We initialize *half* of the weights randomly with Gaussian and *freeze* them, then optimize the rest with back propagation to make the network's output the same as input. After convergence, the weights are our stochastic identity initialization. Other forms of identity initialization involve manually specifying the kernel weights, e.g. [20], which severely skews the distribution of weights (mostly 0s and some 1s). We found that to lead to slower converge and poorer results.

Loss: We use a perceptual loss [45] defined as:

$$D(I_1, I_2) = \sum_l \lambda_l ||\Psi_l(I_1) - \Psi_l(I_2)||_1 + \gamma \sum_{i,j} ||\bar{I}_{1_{i,j}} - \bar{I}_{2_{i,j}}||_1.$$

For Ψ, we use a pretrained VGG16 [82]. $\Psi_l(I)$ denotes the feature map for input image I at l-th convolutional layer. We used all layers except for output layers. λ_l is a scaling coefficient normalized with the number of elements in the feature map. We found perceptual loss to be inherently lossy w.r.t. color information (different colors were projected on one point). Therefore, we add a term to enforce matching statistical moments of color distribution. $\bar{I}_{i,j}$ is the average color vector of a 32×32 tile of the image which is enforced to be matching between I_1 and I_2 using L1 distance and γ is a mixture hyperparameter. We found our final setup to produce superior rendering results to GAN based losses (consistent with some recent works [19]).

3.2.1 Closing the Gap with Real-World: *Goggles*

With all of the imperfections in 3D inputs and point cloud renderings, it is implausible to achieve a fully photo-realistic rendering with neural network fixes. Thus a domain gap with real images would remain. Therefore, we instead formulate the rendering problem as forming a joint space ensuring a correspondence between rendered and real images, and consequently, dissolving the gap [77].

If one wishes to create a mapping between domain S and domain T using a function f, usually a loss with the following form is optimized:

$$\mathcal{L} = \mathbb{E}\left[D(f(\mathcal{I}_s), \mathcal{I}_t)\right]. \tag{1}$$

However, in our case the mapping from T (real images) to S (renderings) is not bijective, or at least the two directions $S \mapsto T$ and $T \mapsto S$ do not appear to be equally difficult. For example, there is no unique solution to dis-occlusion filling, so the domain gap cannot reach zero exercising only $S \mapsto T$ direction. Hence, we add another network u to jointly utilize $T \mapsto S$ and define the objective to be minimizing the distance between $f(\mathcal{I}_s)$ and $u(\mathcal{I}_t)$. Function u is trained to alter the image taken in real-world, $\mathcal{I}_t$, to look like the corresponding rendered image after passing through f network: $f(\mathcal{I}_s)$. Function u can be viewed as corrective glasses of the agent, and hence, the naming of *Goggles*.

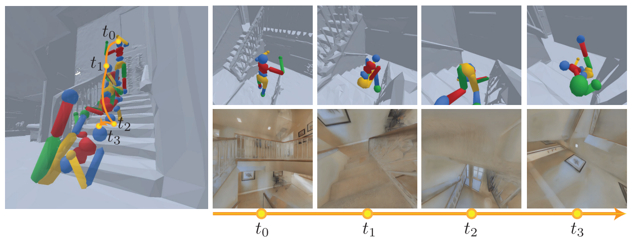

Figure 4: Physics Integration and Embodiment. A Mujoco humanoid model is dropped onto a stairway demonstrating a physically plausible fall along with the corresponding visual observations by the humanoid's eye. The first and second rows show the physics engine view of 4 sampled time steps and their corresponding rendered RGB views, respectively.

To avoid the trivial solution of all images collapsing to a single point, we add the first term in the final loss to enforce preserving a one-to-one mapping. The final loss for training networks u and f is:

$$\mathcal{L} = \mathbb{E}\left[D(f(\mathcal{I}_s), \mathcal{I}_t)\right] + \mathbb{E}\left[D(f(\mathcal{I}_s), u(\mathcal{I}_t))\right]. \quad (2)$$

See Fig. 3 for a visual example. D is the distance defined in Sec 3.2. We use the same network architecture for f and u.

3.3. Embodiment and Physics Integration

Perception and physical constraints are closely related. For instance, the perception model of a human-sized agent should seamlessly develop the notion that it does not fit in the gap under the door and hence should not attend such areas when solving a navigation task; a mouse-sized agent though could fit and its perception should attend such areas. It is thus important for the agent to be constantly subject to constraints of space and physics, e.g. collision, gravity, friction, throughout learning.

We integrated Gibson with a physics engine based on Bullet Physics [24] which supports rigid body and soft body simulation with discrete and continuous collision detection. We also use Bullet's built-in fast collision handling system to record agent's certain interactions, such as how many times it collides with physical obstacles. We use Coulomb friction model by default, as scanned models do not come with material property annotations and certain physics aspects, such as friction, cannot be directly simulated.

Agents: Gibson supports importing arbitrary agents with URDFs. Also, a number of agents are integrated as entry points, including humanoid and ant of Roboschool [4, 75], husky car [1], drone, minitaur [3], Jackrabbot [2]. Agent models are in ROS or Mujoco XML format.

Integrated Controllers: To enable (optionally) abstracting away low-level control and robot dynamics for the tasks that are wished to be approached in a more high-level manner, we also provide a set of practical and ideal controllers to deduce the complexity of learning to control from scratch. We integrated a PID controller and a Nonholonomic controller as well as an ideal positional controller which completely abstracts away agent's motion dynamics.

3.4. Additional Modalities

Besides rendering RGB images, Gibson provides additional channels, such as depth, surface normals, and semantics. Unlike RGB images, these channels are more robust to noise in input data and lighting changes, and we render them directly from mesh files. Geometric modalities, e.g. depth, are provided for all models and semantics are available for 52,561 m^2 of area with semantic annotations from 2D-3D-S [9] and Matterport3D [16] datasets.

Similar to other robotic simulation platforms, we also provide configurable proprioceptive sensory data. A typical proprioceptive sensor suite includes information of joint positions, angle velocity, robot orientation with respect to navigation target, position and velocity. We refer to this typical setup as "non-visual sensor" to distinguish from "visual" modalities in the rest of the paper.

4. Tasks

Input-Output Abstraction: Gibson allows defining arbitrary tasks for an agent. To provide a common abstraction for this, we follow the interface of OpenAI Gym [13]: at each timestep, the agent performs an action at the environment; then the environment runs a forward step (integrated with the physics engine) and returns the accordingly rendered visual observation, reward, and termination signal. We also provide utility functions to keyboard operate an agent or visualize a recorded run.

4.1. Experimental Validation Tasks

In our experiments, we use a set of sample active perceptual tasks and static-recognition tasks to validate Gibson. The active tasks include:

Local Planning and Obstacle Avoidance: An agent is randomly placed in an environment and needs to travel to a random nearby target location provided as relative coordinates (similar to flag run [4]). The agent receives no information about the environment except a continuous stream of depth and/or RGB frames and needs to plan perceptually (e.g. go around a couch to reach the target behind).

Distant Visual Navigation: Similar to the the previous task, but the target location is significantly further away and fixed. Agent's initial location is still randomized. This is similar to the task of auto-docking for robots from a distant location. Agent receives no external odometry or GPS information, and needs to form a contextual map to succeed.

Stair Climb: An (ant [4]) agent is placed on on top of a stairway and the target location is at the bottom. It needs to learn a controller for its complex dynamics to plausibly go down the stairway without flipping, using visual inputs.

To benchmark how close to real images the renderings of Gibson are, we used two static-recognition tasks: depth estimation and scene classification. We train a neural network using $(rendering, ground\ truth)$ pairs as training

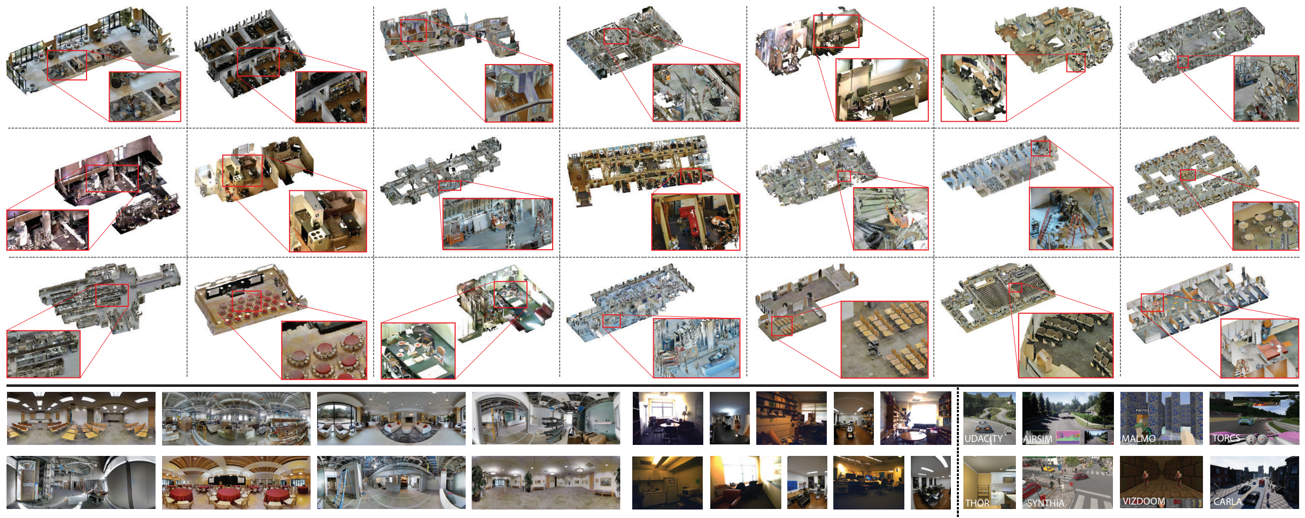

Figure 5: Sample spaces in Gibson database. The spaces are diverse in terms of size, visuals, and function, e.g. businesses, construction sites, houses. Upper: Sample 3D models. Lower: Sample images from Gibson database (left) and some of other environments [29, 46, 67, 79, 48, 94, 35, 100] (right).

data, but test them on $(real\ image, ground\ truth)$. If Gibson renderings are close enough to real images and *Goggles* mechanism is effective, test results on real images are expected to be satisfactory. This also enables quantifying the impact of *Goggles*, i.e. using $u(\mathcal{I}_t)$ vs. $\mathcal{I}_s$, $f(\mathcal{I}_s)$, and $\mathcal{I}_t$.

Depth Estimation: Predicting depth given a single RGB image, similar to [31]. We train 4 networks to predict the depth given one of the following 4 as input images: $\mathcal{I}_s$ (pre-neural network rendering),$f(\mathcal{I}_s)$ (post-neural network rendering), $u(\mathcal{I}_t)$ (real image seen with *Goggles*), and $\mathcal{I}_t$ (real image). We compare the performance of these in Sec. 5.3.

Scene Classification: The same as previous task, but the output is scene classes rather than depth. As our images do not have scene class annotations, we generate them using a well performing network trained on Places dataset [98].

5. Experimental Results

5.1. Benchmarking Space Databases

The spaces in Gibson database are collected using various scanning devices, including NavVis, Matterport, or DotProduct, covering a diverse set of spaces, e.g. offices, garages, stadiums, grocery stores, gyms, hospitals, houses. All spaces are fully reconstructed in 3D and post processed to fill the holes and enhance the mesh. We benchmark some of the existing synthetic and real databases of spaces (SUNCG [84] and Matterport3D [16]) vs Gibson's using the following metrics in Table 1:

Specific Surface Area (SSA): the ratio of inner mesh surface and volume of convex hull of the mesh. This is a measure of clutter in the models.

Navigation Complexity: Longest A^* navigation distance between randomly placed two points divided by the straight line distance. We compute the highest navigation complexity $\max_{s_i,s_j} \frac{d_{A^*}(s_i,s_j)}{d_{l2}(s_i,s_j)}$ for every model.

Dataset	Gibson	SUNCG	Matterport3D
Number of Spaces	572	45622	90
Total Coverage m^2	211k	5.8M	46.6K
SSA	1.38	0.74	0.92
Nav. Complexity	5.98	2.29	7.80
Real-World Transfer Err	$0.92^{\S}$	$2.89^{\dagger}$	$2.11^{\dagger}$

Table 1: Benchmarking Space Databases: Comparison of Gibson database with SUNCG [84] (hand designed synthetic), and Matterport3D [16]. § Rendered with Gibson, † rendered with MINOS [72].

Real-World Transfer Error: We train a neural network for depth estimation using the images of each database and test them on real images of 2D-3D-S dataset [9]. Training images of SUNCG and Matterport3D are rendered using MINOS [72] and our dataset is rendered using Gibson's engine. The training set of each database is 20k random RGB-depth image pairs with $90°$ field of view. The reported value is average depth estimation error in meters.

Scene Diversity: We perform scene classification on 10k randomly picked images for each database using a network pretrained on [98]. We report the entropy of the distribution of top-1 classes for each environment. Gibson, SUNCG [84], and THOR [100] gain the scores of 3.72, 2.89, and 3.32, respectively (highest possible entropy = 5.90).

5.2. Evaluation of View Synthesis

To train the networks f and u of our neural network based synthesis framework, we sampled 4.3k 1024×2048 $\mathcal{I}_s$—$\mathcal{I}_t$ panorama pairs and randomly cropped them to 256×256. We use Adam [51] optimizer with learning rate 2×10^{-4}. We first train f for 50 epochs until convergence, then we train f and u jointly for another 50 epochs with learning rate 2×10^{-5}. The learning finishes in 3 days on 2 Nvidia Titan X GPUs.

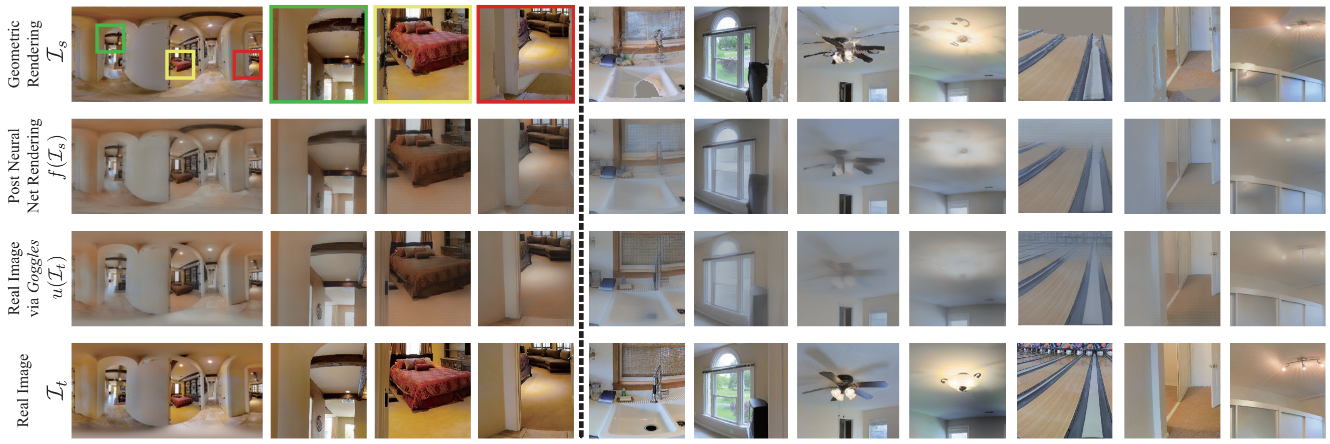
Geometric Rendering $\mathcal{I}_s$
Post Neural Net Rendering $f(\mathcal{I}_s)$
Real Image via *Goggles* $u(\mathcal{I}_t)$
Real Image $\mathcal{I}_t$

Figure 6: Qualitative results of view synthesis and *Goggles*. Top to bottom rows show images before neural network correction, after neural network correction, target image seen through *Goggles*, and target image (i.e. ground truth real image). The first column shows a pano and the rest are sample zoomed-in patches. Note the high similarity between 2^{nd} and 3^{rd} row, signifying the effectiveness of *Goggles*.

Resolution	128x128	256x256	512x512
Non-Visual Sensor	427.9	427.9	427.9
Depth Only	159.4	113.3	79.2
RGBD Pre Network f	81.5	50.9	33.3
RGBD Post Network f	73.6	42.7	18.3
Semantic Only	93.1	79.5	50.9
Surface Normal	89.3	73.7	45.4

Table 2: Rendering speed (FPS) of Gibson for different resolutions and configurations. Tested on a single NVIDIA GeForce GTX1070 card.

Train	Test	Static Tasks		Appearance	
		Scene Class Acc.	Depth Est. Error	SSIM	L1
$\mathcal{I}_s$	$\mathcal{I}_t$	0.280	1.026	0.627	0.096
$f(\mathcal{I}_s)$	$\mathcal{I}_t$	0.266	1.560	0.480	0.10
$f(\mathcal{I}_s)$	$u(\mathcal{I}_t)$	**0.291**	**0.915**	**0.816**	**0.051**

Table 3: Evaluation of view synthesis and transferring to real-world. *Static Tasks* column shows on both scene classification task and depth estimation tasks, it is easiest to transfer from $f(\mathcal{I}_s)$ to $u(\mathcal{I}_t)$ compared with other cross-domain transfers. *Appearance* columns compare L1 and SSIM distance metrics for different pairs showing the combination of network f and *Goggles* u achieves best results.

Sample renderings and their corresponding real image (ground truth) are shown in Fig. 6. Note that pre-neural network renderings suffer from geometric artifacts which are partially resolved in post-neural network results. Also, though the contrast of the post-neural network images is lower than real ones and color distributions are still different, *Goggles* could effectively alter the real images to match the renderings (compare 2^{nd} and 3^{rd} rows). In additional, the network f and *Goggles* u jointly addressed some of the pathological domain gaps. For instance, as lighting fixtures are often thin and shiny, they are not well reconstructed in our meshes and usually fail to render properly. Network f and *Goggles* learned to just suppress them altogether from images to not let a domain gap remain. The scene out the windows also often have large re-projection errors, so they are usually turned white by f and u.

Appearance columns in Table 3 quantify view synthesis results in terms image similarity metrics L1 and SSIM. They echo that the smallest gap is between $f(\mathcal{I}_s)$ and $u(\mathcal{I}_t)$.

Rendering Speed of Gibson is provided in Table 2.

5.3. Transferring to Real-World

We quantify the effectiveness of *Goggles* mechanism in reducing the domain gap between Gibson renderings and real imagery in two ways: via the static-recognition tasks described in Sec. 4.1 and by comparing image distributions.

Evaluation of transferring to real images via scene classification and depth estimation are summarized in Table. 3. Also, Fig. 7 (a) provides depth estimation results for all feasible train-test combinations for reference. The diagonal values of the 4×4 matrix represent training and testing on the same domain. The gold standard is train and test on $\mathcal{I}_t$ (real images) which yields the error of 0.86. The closest combination to that in the entire table is train on $f(I_s)$ (f output) and test on $u(I_t)$ (real image through *Goggles*) giving 0.91, which signifies the effectiveness of *Goggles*.

In terms of distributional quantification, we used two metrics of Maximum Mean Discrepancy (MMD) [37] and CORAL [86] to test how well $f(\mathcal{I}_s)$ and $u(\mathcal{I}_t)$ domains are aligned. The metrics essentially determine how likely it is for two samples to be drawn from different distributions. We calculate MMD and CORAL values using the features of the last convolutional layer of VGG16 [82] and kernel $k(x, y) = x^T y$. Results are summarized in Fig. 7 (b) and (c). For each metric, $f(\mathcal{I}_s)$ - $u(\mathcal{I}_t)$ is smaller than other pairs, showing that the two domains are well matching.

In order to quantitatively show the networks f and u do not give degenerate solutions (i.e. collapsing all images to few points to close the gap by cheating), we use $f(\mathcal{I}_s)$ and $u(\mathcal{I}_t)$ as queries to retrieve their nearest neighbor using VGG16 features from $\mathcal{I}_s$ and $\mathcal{I}_t$, respectively. Top-1, 2 and 5 accuracies for $f(\mathcal{I}_s) \mapsto \mathcal{I}_s$ are 91.6%, 93.5%, 95.6%. Top-1, 2 and 5 accuracies for $u(\mathcal{I}_t) \mapsto \mathcal{I}_t$ are 85.9%, 87.2%,89.6%. This indicates a good correspondence be-

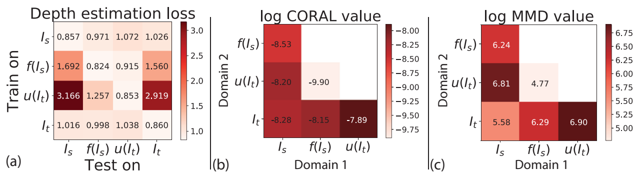

Figure 7: Evaluation of transferring to real-world from Gibson. (a) Error of depth estimation for all train-test combinations. (b,c) MMD and CORAL distributional distances. All tests are in support of *Goggles*.

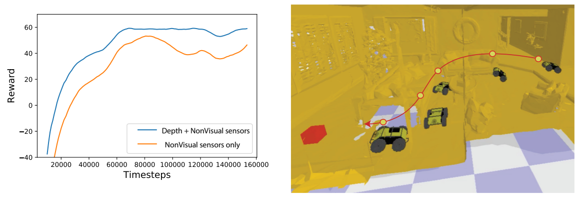

Figure 8: Visual Local planning and obstacle avoidance. Reward curves for perceptual vs non-perceptual husky agents and a sample trajectory.

tween pre and post neural network images is preserved, and thus, no collapse is observed.

5.4. Validation Tasks Learned in Gibson

The results of the active perceptual tasks discussed in Sec. 4.1 are provided here. In each experiment, the non-visual sensor outputs include agent position, orientation, and relative position to target. The agents are rewarded by the decrease in their distance towards their targets. In Local Planning and Visual Obstacle Avoidance, they receive an additional penalty for every collision.

Local Planning and Visual Obstacle Avoidance Results: We trained a perceptual and non-perceptual husky agent according to the setting in Sec. 4.1 with PPO [74] for 150 episodes (300 iterations, 150k frames). Both agents have a four-dimensional discrete action space: forward/backward/left/right. The average reward over 10 iterations are plotted in Fig 8. The agent with perception achieves a higher score and developed obstacle avoidance behavior to reach the goal faster.

Distant Visual Navigation Results: Fig. 9 shows the target and sample random initial locations as well as the reward curves. Global navigation behavior emerges after 1700 episodes (680k frames), and only the agent with visual state was able to accomplish the task. The action space is the same as previous experiment.

Also, we use the trained policy of distant navigation to evaluate the impact of *Goggles* on an active task: we go to camera locations where $\mathcal{I}_t$ is available. Then we measure the policy discrepancy in terms of L2 distance of output action logits when different renderings and $\mathcal{I}_t$ are provided as input. Training on $f(\mathcal{I}_s)$ and testing on $u(\mathcal{I}_t)$ yields discrepancy of 0.204 (best), while training on $f(\mathcal{I}_s)$ and testing on $\mathcal{I}_t$ gives 0.300 and training on $\mathcal{I}_s$ and testing on $\mathcal{I}_t$ gives 0.242. After the initial release of our work, a paper recently reported an evaluation done on a real robot for adaptation using inverse mapping from real images to renderings [97], with positive results. They did not use paired data, unlike Gibson, which would be expected to further enhance the results.

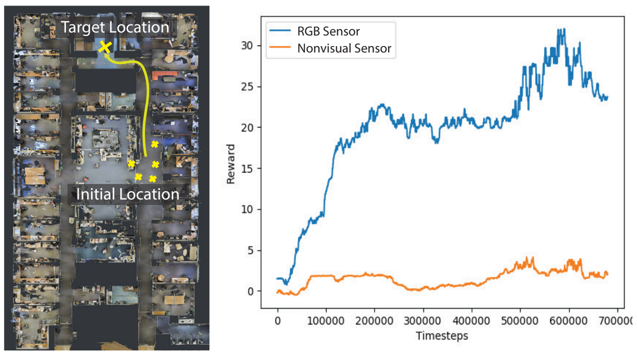

Figure 9: Distant Visual Navigation. The initial locations and target are shown. The agent succeeds only when provided with visual inputs.

Stair Climb: As explained in Sec. 4.1, an ant [4] is trained to perform the complex locomotive task of plausibly climbing down a stairway without flipping. The action space is eight dimensional continuous torque values. We train one perceptual and one non-perceptual agent starting at a fixed initial location, but at test time slightly and randomly move their initial and target location around. They start to acquire stair-climbing skills after 1700 episodes (700k time steps). While the perceptual agent learned slower, it showed better generalizability at test time coping with the location shifts and outperformed the non-perceptual agent by 70%. Full details of this experiment is privded in the supplementary material.

6. Limitations and Conclusion

We presented Gibson Environments for developing real-world perception for active agents and validated it using a set of tasks. While we think this is a step forward, there are some limitations that should be noted. First, though Gibson provides a good basis for learning complex navigation and locomotion, it does not include dynamic content (e.g. other moving objects) and does not allow manipulation at this point. This can potentially be solved by integrating our approach with synthetic objects [17, 47]. Second, we do not have full material properties and no existing physics simulator is optimal; this may lead to physics related domain gaps. Finally, we provided quantitative evaluations of *Goggles* mechanism for transferring to real world mostly using static recognition tasks. The ultimate test would be evaluating *Goggles* on real robots.

Acknowledgement: We gratefully acknowledge the support of Facebook, Toyota (1186781-31-UDARO), ONR MURI (N00014-14-1-0671), ONR (1165419-10-TDAUZ); Nvidia, CloudMinds, Panasonic (1192707-1-GWMSX).

References

[1] Husky UGV - Clearpath Robotics. http://wiki.ros.org/Robots/Husky. Accessed: 2017-09-30. 5

[2] Jackrabbot - Stanford Vision and Learning Group. http://cvgl.stanford.edu/projects/jackrabbot/. Accessed: 2018-01-30. 5

[3] Legged UGVs - Ghost Robotics. https://www.ghostrobotics.io/copy-of-robots. Accessed: 2017-09-30. 5

[4] OpenAI Roboschool. http://blog.openai.com/roboschool/. Accessed: 2018-02-02. 5, 8

[5] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010. 2

[6] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007. 2

[7] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016. 2

[8] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[9] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017. 3, 5, 6

[10] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013. 2

[11] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006. 2

[12] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994. 2

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 5

[14] R. A. Brooks. Elephants don't play chess. *Robotics and autonomous systems*, 6(1-2):3–15, 1990. 1, 2

[15] R. A. Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991. 1

[16] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 3, 5, 6

[17] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 8

[18] C.-F. Chang, G. Bishop, and A. Lastra. Ldi tree: A hierarchical representation for image-based rendering. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 291–298. ACM Press/Addison-Wesley Publishing Co., 1999. 3

[19] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *arXiv preprint arXiv:1707.09405*, 2017. 4

[20] Q. Chen, J. Xu, and V. Koltun. Fast image processing with fully-convolutional networks. In *IEEE International Conference on Computer Vision*, 2017. 4

[21] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288. ACM, 1993. 3

[22] P. S. Churchland, V. S. Ramachandran, and T. J. Sejnowski. A critique of pure vision. *Large-scale neuronal theories of the brain*, pages 23–60, 1994. 1

[23] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun. End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*, 2017. 2

[24] E. Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 15:49, 2013. 5

[25] H. Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009. 2

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 2

[27] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE transactions on Robotics and Automation*, 17(6):905–908, 2001. 2

[28] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016. 2

[29] 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. 2, 6

[30] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1538–1546. IEEE, 2015. 3

[31] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 6

[32] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[33] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 512–519. IEEE, 2016. 2

[34] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 2

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 6

[36] J. J. Gibson. *The ecological approach to visual perception*. Psychology Press, 2013. 1

[37] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012. 7

[38] A. Gupta. Supersizing self-supervision: Learning perception and action without human supervision. 2016. 2

[39] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[40] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):231, 2016. 3

[41] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017. 2

[42] R. Held and A. Hein. Movement-produced stimulation in the development of visually guided behavior. *Journal of comparative and physiological psychology*, 56(5):872, 1963. 1

[43] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese. Gonet: A semi-supervised deep learning approach for traversability estimation. *arXiv preprint arXiv:1803.03254*, 2018. 2

[44] C. Jiang, Y. Zhu, S. Qi, S. Huang, J. Lin, X. Guo, L.-F. Yu, D. Terzopoulos, and S.-C. Zhu. Configurable, photorealistic image rendering and ground truth synthesis by sampling stochastic grammars representing indoor scenes. *arXiv preprint arXiv:1704.00112*, 2017. 2

[45] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 4

[46] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247, 2016. 2, 6

[47] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30, page 157. ACM, 2011. 8

[48] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016. 2, 6

[49] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986. 2

[50] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. 2

[51] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[52] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015. 3

[53] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[54] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. 2

[55] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, page 0278364917710318, 2016. 2

[56] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996. 3

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[58] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 7–ff. ACM, 1997. 3

[59] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3521–3529, 2016. 2

[60] R. Ortiz-Cayon, A. Djelouah, and G. Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *International Conference on 3D Vision-3DV*, 2015. 3

[61] A. R. Parker. On the origin of optics. *Optics & Laser Technology*, 43(2):323–329, 2011. 1

[62] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363*, 2017. 2

[63] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016. 2

[64] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 2

[65] 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. 2

[66] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[67] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A large collection

of synthetic images for semantic segmentation of urban scenes. 2016. 6

[68] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016. 2

[69] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011. 2

[70] F. Sadeghi and S. Levine. rl: Real singleimage flight without a single real image. arxiv preprint. *arXiv preprint arXiv:1611.04201*, 12, 2016. 2

[71] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 2

[72] 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. 6

[73] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015. 2

[74] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 8

[75] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *ArXiv e-prints*, July 2017. 5

[76] S. M. Seitz and C. R. Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30. ACM, 1996. 3

[77] O. Sener, H. O. Song, A. Saxena, and S. Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016. 2, 4

[78] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242. ACM, 1998. 3

[79] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. 2, 6

[80] E. Shechtman, A. Rav-Acha, M. Irani, and S. Seitz. Regenerative morphing. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 615–622. IEEE, 2010. 3

[81] H. A. Simon. *The sciences of the artificial*. MIT press, 1996. 2

[82] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4, 7

[83] L. Smith and M. Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005. 1

[84] 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. 2, 6

[85] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, volume 6, page 8, 2016. 2

[86] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops*, pages 443–450. Springer, 2016. 7

[87] S. Suwajanakorn, I. Kemelmacher-Shlizerman, and S. M. Seitz. Total moving face reconstruction. In *European Conference on Computer Vision*, pages 796–812. Springer, 2014. 3

[88] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016. 3

[89] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017. 2

[90] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *European Conference on Computer Vision*, pages 836–850. Springer, 2014. 3

[91] D. M. Wolpert and Z. Ghahramani. Computational principles of movement neuroscience. *Nature neuroscience*, 3:1212–1217, 2000. 1

[92] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018. 2

[93] M. Wulfmeier, A. Bewley, and I. Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. *arXiv preprint arXiv:1703.01461*, 2017. 2

[94] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4, 2000. 6

[95] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint arXiv:1612.01079*, 2016. 2

[96] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 4

[97] J. Zhang, L. Tai, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard. Vr goggles for robots: Real-to-sim domain adaptation for visual control. *arXiv preprint arXiv:1802.00265*, 2018. 8

[98] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 6

[99] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer, 2016. 3

[100] 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. 2, 6