Supplementary Material: Deep Visual Teach and Repeat on Path Networks

Tristan Swedish
MIT Media Lab
tswedish@mit.edu

Ramesh Raskar
MIT Media Lab
raskar@mit.edu

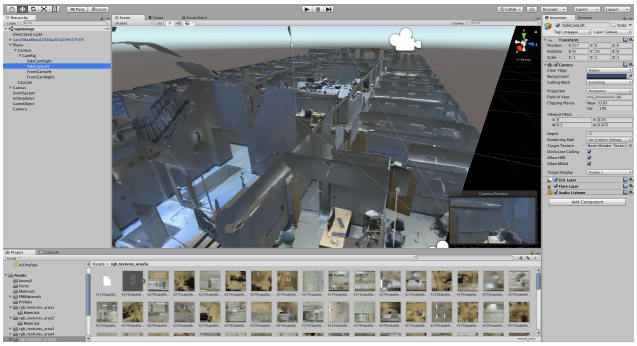

Figure 1: Project view of the Unity evaluation environment.

Figure 2: Captured frame from the evaluation environment. The four panes are different cameras in the scene, enabling multiple simultaneous viewpoints from the same agent. The bar at the bottom of the screen prints ground truth values that can easily be parsed by an OCR system.

1. Virtual Evaluation Environment

We make use of a virtual evaluation environment in order to produce most of the results in the paper. The evaluation environment was created in Unity, and consists of three main components independent of the S3-3D-2D dataset [1]. We imported the textured mesh elements of the S3-3D-2D dataset into Unity, developed an agent centric view and added gravity and collision physics to the environments, and then recorded ground truth pose information so that observations could later be labeled.

S3-3D-2D Rendering The use of the Unity Game Engine simplifies rendering of the textured mesh provided by the S3-3D-2D. We utilized the standard global illumination provided by the engine as well as a parallel ray light source direction above the environment. This source casts shadows by the scene geometry, and was configured directly above the scene to simulate overhead lights in the scene. Additional illumination could be added to the scene using Unity's tools, but we elected to use this one additional illumination source for our experiments.

We also added a "Mesh Collider" to the imported mesh so that any objects running in the physics engine would properly collide with the surface. This ensured that the game agent would not move through walls and that positioning was properly simulated even if movement controls indicated motion that may be impossible. This aspect of the environment is critical for future closed loop simulation, but also for ensuring collected data is as true to real data capture as possible.

Agent View and Control Each observation was generated by a camera in the Unity engine with an idealized pinhole camera with intrinsic camera parameters matched to the Galaxy Edge S6+ smartphone (with 46 degree vertical field of view). Since the neural network in these experiments only required an input resolution of 224×224, multiple camera views were rendered in Unity at a resolution of 1920×1080, enabling four simultaneous cameras to be rendered each frame. Each of these camera frames represents deviations from the intended agent motion so that robustness to geometrical offsets can be easily examined.

In the virtual environment, the agent has two axis of control: forward/backward and rotate left/right. Each control input accelerates the agent to a maximum velocity along each control axis, so movement is not instantaneous. The maximum velocity was 0.5 m/s. For the purposes of data collection in the paper, frame data was captured at 30 fps us-

ing a screen capture utility available in Ubuntu 16.04 called "Kazam."

Labeling and ground-truth Extraction After capturing frame data with Kazam, each region of interest in each frame was extracted for later processing. First, the observation pane corresponding to the desired monocular view was extracted from the frame by cropping. Second, the bottom ground truth pane was extracted and passed through the Tesseract OCR engine [2]. The reason ground truth was printed directly on each frame was to account for any latency that may occur in any frame buffer extraction method used by Kazam or other virtual screen capture methods. Practically, it was difficult to ensure synchronization between captured frame timestamps and actual game state that could be extracted from the Unity engine through alternative means (i.e. printed in a log).

2. Simulated Path Detail

Validation Routes The S3-3D-2D dataset consists of six different indoor environments. Following one of the suggested data splits, our validation set contains data from Areas 1,3, and 6. Since our method does not use any explicit training data outside of its testing environment, all data examined in this paper use data collected from these areas. Future work may utilize training data from the other areas, facilitating standardized comparisons in the future.

Path following data was generated for each test area by manually directing an agent through 6 paths. The agent was driven along each path twice, the first path corresponds to the "training" path used to generate the Path Following data, and the second path was used to query the prediction. In all cases the accuracy for predictions from the query path was determined using ground truth. In total, this corresponded to 18 unique paths for 36 total paths between three areas in the S3-3D-2D dataset.

We show the paths used in Figure.

Lighting Perturbations In the experiments describing lighting perturbations, the directional light in each scene was modified to use a different RGB code and total scaling factor (available as parameters in the Unity engine). These changes are supposed to simulate overhead lights both on and off, leaving only ambient global illumination. Below we enumerate the lighting changes corresponding to these states:

- **Day (normal experiments)**: R:255 G:250 B:229 Scale: 0.8
- **Dusk**: R: 255 G: 220 B: 193 Scale: 0.5
- **Night**: R: 128 G: 150: B: 178 Scale: 0.0

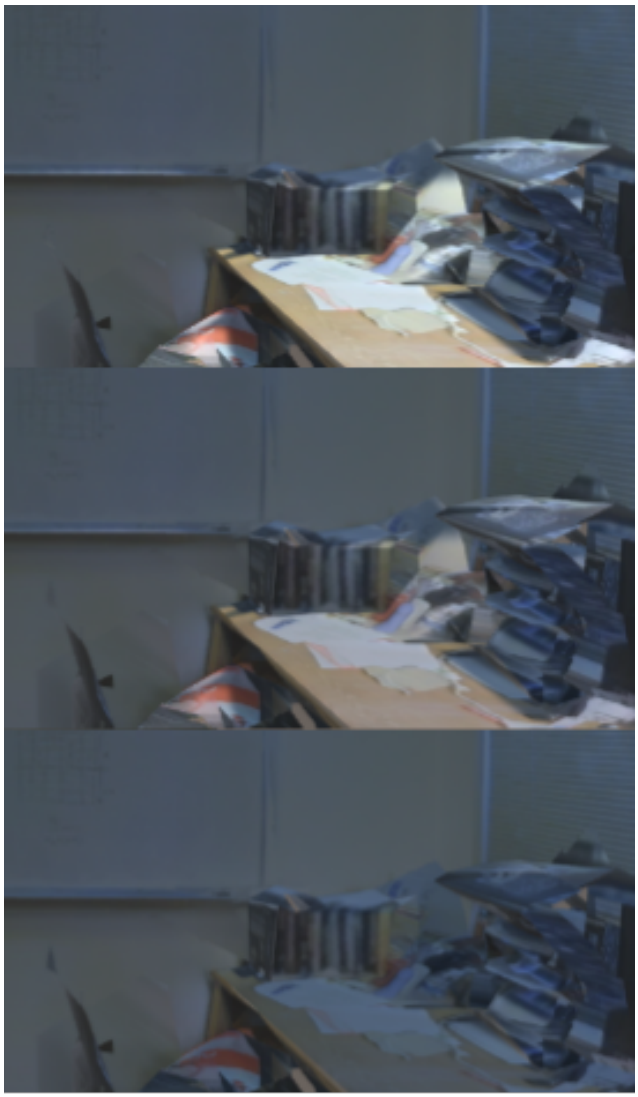

Figure 3: Three different lighting condition used to show invariance to lighting conditions. Top: Day, Middle: Dusk, Bottom: Night.

The lighting difference is apparent in Figure 3, showing how the perceived image changes dramatically for objects in the scene that would normally be illuminated by overhead lights. Objects primarily lit by global illumination (such as the back wall), do not change as dramatically. This kind of lighting perturbation is more consistent with real-world scenes, and does not correspond to a simple channel-wise gain change over the entire image.

3. Path-Following Evaluation Ground Truth

We show all the ground truth paths for each of the tested paths in the virtual evaluation environment in Figure 4.

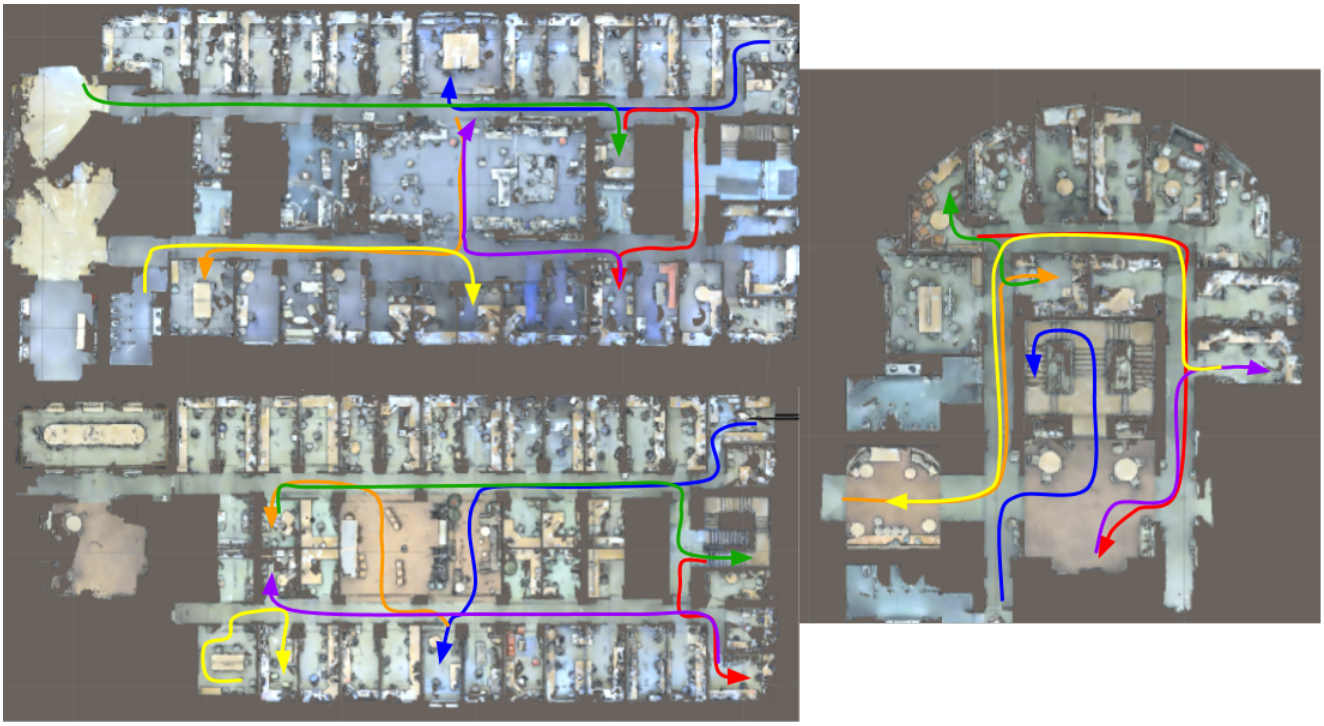

Figure 4: The evaluation paths used for the three areas. Top Left: Area 1, Right: Area 3, Bottom Left: Area 6. Color to index order is: blue, orange, green, red, purple, yellow.

References

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

[2] R. Smith, D. Antonova, and D.-S. Lee. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, MOCR '09, pages 1:1–1:8, New York, NY, USA, 2009. ACM.