Episodic Memory Question Answering (Supplementary Document)

1. Additional Dataset Details

The Matterport3D [3] meshes are annotated with 40 object categories. Following prior work [2], we work with the 12 most commonly occurring object types, leaving out objects such as walls, curtains that are rendered as a thin array of pixels on the top-down map. The list of the 12 object categories used in our work is as follows: shelving, fireplace, bed, table, plant, drawers, counter, cabinet, cushion, sink, sofa and chair.

Fig. 1 shows additional qualitative examples of questions and associated ground-truth, top-down answer maps from our dataset.

We also show additional analysis of the distribution of locations and sizes of objects from our questions (Fig. 2 and 3). Fig. 2 visualizes the spatial distribution of 6 object types (drawers, sink, plant, fireplace, cushion and shelving) over the 250×250 spatial maps across the "short tours" data splits (recall that this is the dataset that we train our models on). Each object instance is represented by a circle with the center of the circle corresponding to the location of an object instance and the radius representing its size (on the top-down map). The sizes of object instances are measured in terms of the number of pixels. As mentioned in Sec. 3 (main paper), all top-down maps are of a fixed 2cm $\times$2cm resolution, therefore the number of pixels covered by an object on the top-down map has a direct correlation with its size in the real world.

As evident from Fig. 2, the objects demonstrate a good distribution over spatial locations in the map, thereby alleviating dataset biases such as "are sinks almost always found in specific corners of the map?". Recall that, the fact that we beat the LangOnly baseline by almost 150% is also indicative of the fact that our models are free from such biases as well.

Additionally, Fig. 3 (left) shows the mean object locations for all 12 categories across the "short tours" data splits. Notice that, consistent with Fig. 2, the mean locations for nearly all object categories are close to the center of the 250×250 map. This further confirms a comprehensive distribution of objects across all spatial locations in the map. Fig. 3 (right) shows a distribution of average object sizes per category. Here, we can see that our dataset involves asking questions about objects of varying sizes – from typically small objects such as cushions to bigger ones such as beds.

2. Additional Model+Training Details

Details of the temporal features. In this section, we provide more details regarding the construction of the temporal features component of our scene memories. At a high level, we save temporal features at every (i, j) gridcell on the top-down map. To do that, we first represent each video tour (of varying lengths) as a sequence of 20 (determined heuristically) equal-length segments. For every tour segment, we then compute the subset of top-down grid-cells that were observed by the agent during that part of the tour. This induces a binary topdown map, per tour segment, indicating whether a given (i, j) was observed during that part of the tour or not. Aggregated over all segments via channel-wise stacking, we get a 20-dim multi-hot vector at each (i, j). Intuitively, these 20-bits for any top-down location (i, j) indicate the time instances (in terms of segments) during the tour when that metric location was observed.

Details of the LingUNet architecture. Fig. 3 in the main paper shows a schematic diagram of our EMQA agent. We show the same in Fig. 4 for the readers' convenience and provide more details about our LingUNet-based question-answering module in this section. Recall (from Sec. 4 in the main paper) that our LingUNet-based question-answering model takes the constructed scene memory features and an embedding of the question as an input and generates a top-down heatmap of the agent's answer prediction as its output. We use a 3-layer LingUNet architecture, as shown in Fig. 4. The module receives a tensor of dimensions $256 \times 250 \times 250$ (the spatio-temporal memory tensor) and a 64-dim question LSTM embedding as inputs. The scene memory tensor is encoded through a series of three, 3×3 Conv. layers and the question embedding is L2-normalized and used to generate three convolutional filters via FC-layers. These question-conditioned convolutional filters operate on the intermediate feature outputs from the 3 encoder Conv. layers to generate the language-conditioned skip connections. Finally, the encoded feature volumes are passed through a series of 3 de-convolutional layers (see Fig. 4), with added skip

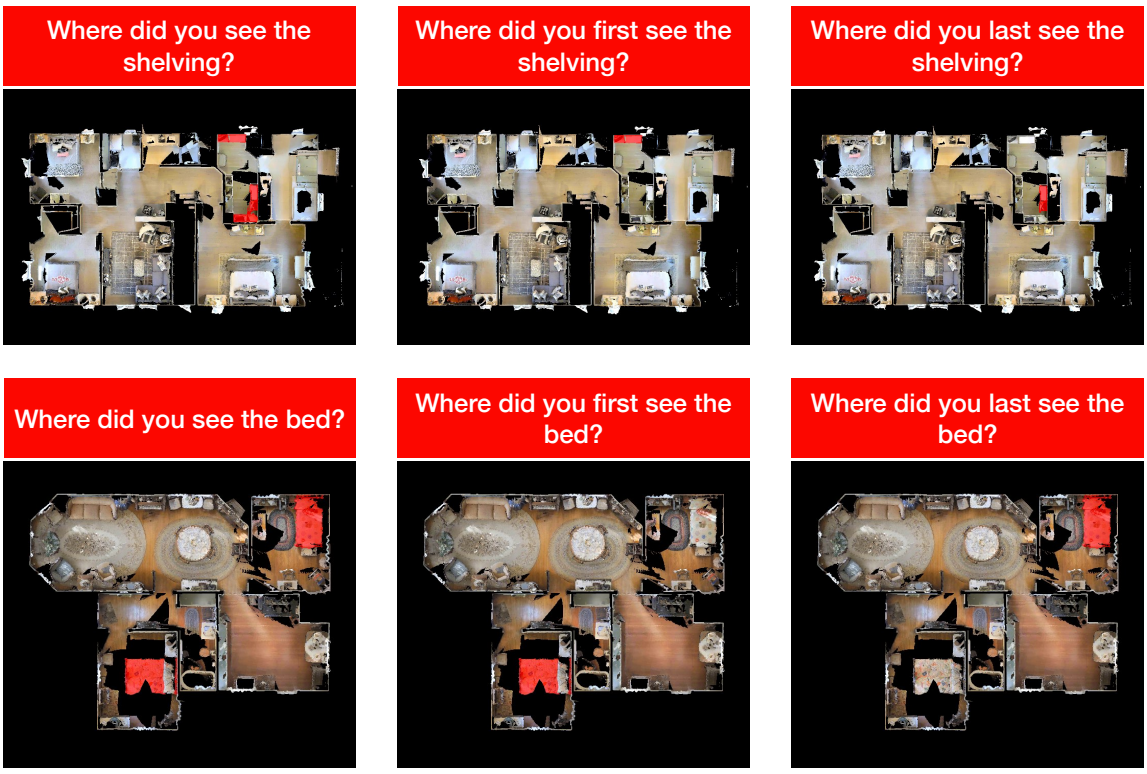

Figure 1. This figure illustrates more examples from our proposed EMQA dataset.

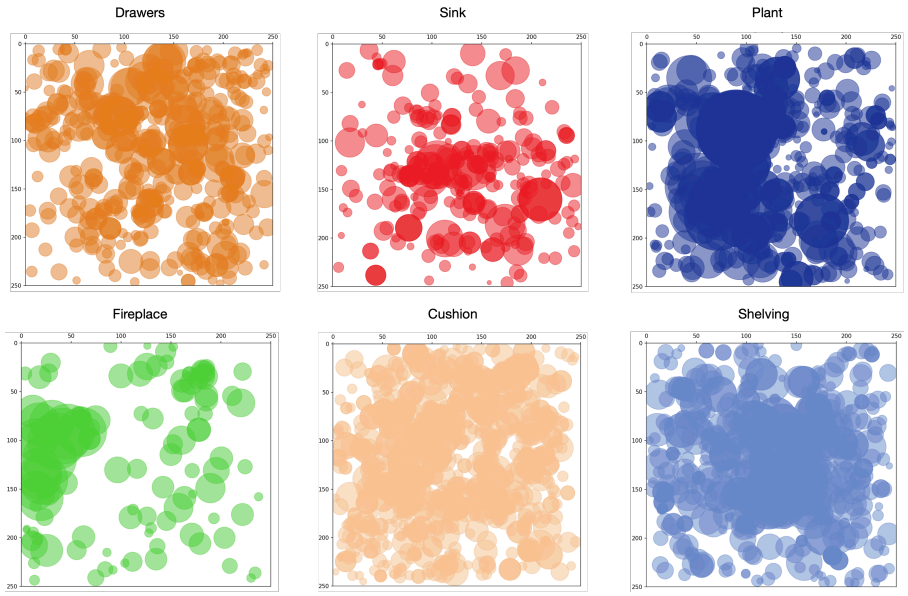

Figure 2. Distribution of spatial arrangement and size of object instances across 6 different object categories. Each instance is represented by a circle whose center and radius correspond to the location and size of the object, as viewed on the top-down map.

connections to generate the final output map: a tensor containing a 128-dim feature vector for each of the 250×250 spatial locations of the input map.

The output feature map from the LingUNet model is post-processed by a 1×1 Conv. layer, followed by Sigmoid non-linearity to generate the agent's prediction score

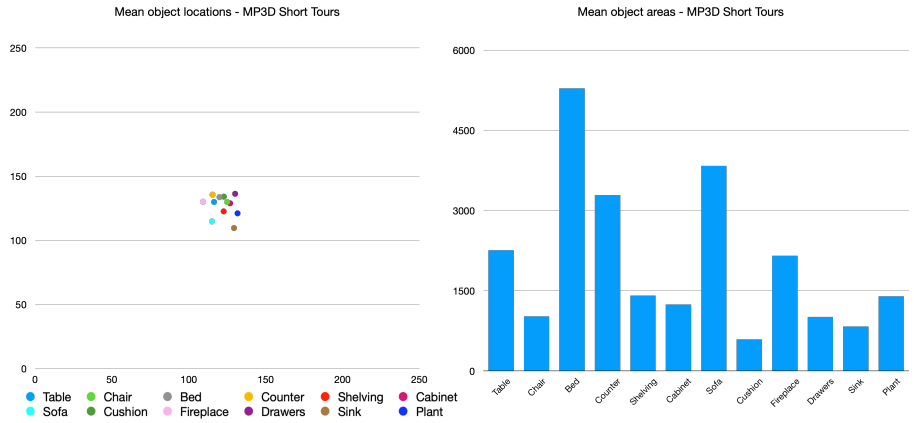

Figure 3. [Left] Scatter-plot showing the mean locations of all instances across a given object category, for all object categories. [Right] A distribution of the mean object size on the top-down map across all categories demonstrating the spectrum of object sizes in our dataset.

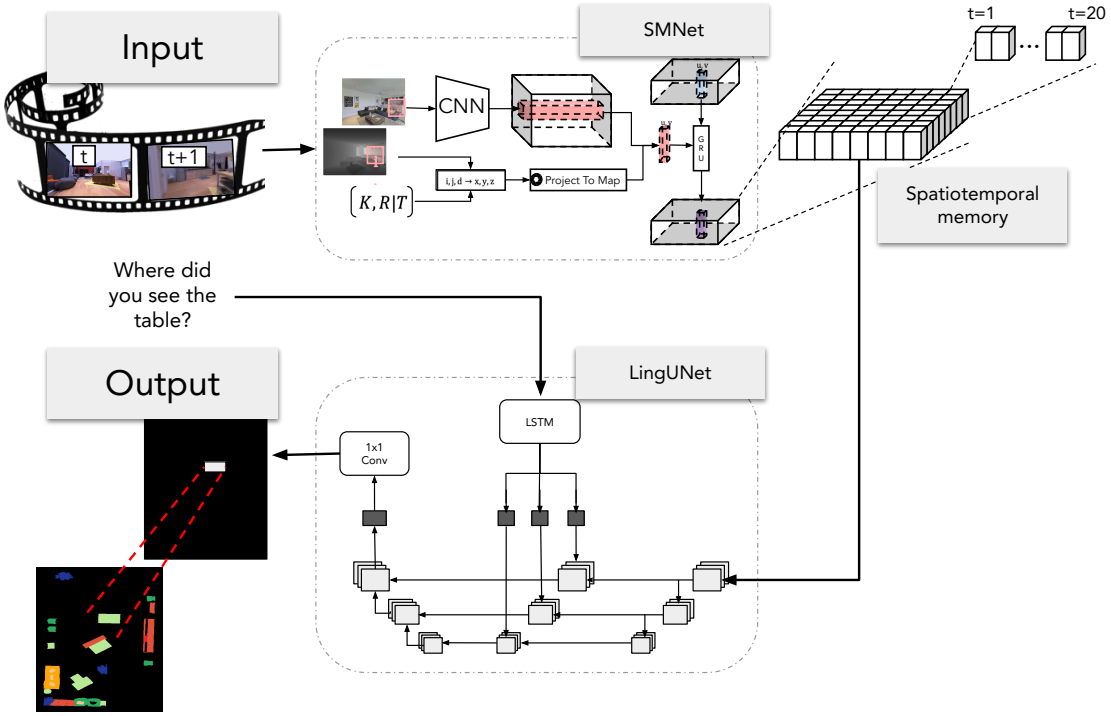

Figure 4. A schematic diagram of our proposed EMQA agent. Our agent first constructs an episodic memory representation of the tour and then grounds answers to questions on the top-down scene floorplan using a LingUNet-based answering model.

for each of the 250×250 spatial map cell.

Other training hyperparameters. As stated in Sec. 4 (main paper), we use a dynamically-weighted variant of the binary cross entropy loss (Focal loss [8]), applied per "pixel" on the top-down map output to train our model to correctly predict whether each metric "pixel" on the map belongs to the answer category or not. We set the gamma parameter in our Focal loss implementation to

2.0 and use the Adam optimizer with a learning rate of $2e{-}4$ and weight-decay of $4e{-}4$. To circumvent memory issues during training, our data loader returns singleton batches of questions (and associated scene memory features). However, we accumulate gradients and update the model weights every $B = 4$ iterations to simulate an effective batch-size of 4.

Details about EgoBuffer baselines. In this section, we provide more architectural details about the EgoBuffer-* baselines (Sec. 5 in the main paper). As mentioned in Sec. 5, as a first step, we compute egocentric feature volumes for every step of the agent tour via a pre-trained RedNet model (the same pre-trained RedNet model weights are used across all our models and baselines).

For the EgoBuffer-Avg baseline, we simply average these per-step, convolutional feature volumes and pass them through a 4-layer convolutional network (followed by a linear layer) to get a flattened, 1-D vector representation of the agent tour.

For the EgoBuffer-GRU baseline, we instead compute the encoded, flattened, 1-D feature representation at every step and then pass them through a GRU. The hidden state from the final time step is the representation of the agent's tour.

And finally, for the EgoBuffer-Attn. baseline, instead of sending the per-step, flattened egocentric features as inputs to a GRU, we perform a scaled, dot-product attention, conditioned on the question-embedding to obtain the tour representation.

Once we obtain the tour representation using the methods described above, we append the features with the question embedding and then decode the multi-modal (question+scene) features into a 250×250 feature map via a 5-layer de-convolutional network. Finally, use a 1×1 Conv + Sigmoid layer to get the answer prediction scores.

Also note that, similar to our proposed model, the family of EgoBuffer-* baselines are trained on fixed-size, 20-step "short" tours. During evaluation on "full tours", we split them into consecutive chunks of 20-steps each, generate a 250×250 output for each such "short tour" segment and then combine them into the full-scale environment map to get the final output (using pose information).

3. Additional Quantitative EMQA Results

As a means to circumvent memory constraints during training and as a data augmentation strategy, we subsample 20-step "short" tours from the full-scale exploration tours in our dataset (see Sec. 3, sub-heading, "Generating "short" tours for training" in the main paper for more details). During training, our model learns to build top-down maps (of smaller spatial dimensions: 250×250 than that of the full scene) and localize answers to questions on the map from these 20-step tours. During evaluation, we test its generalization to building full-scale maps from the entire exploration trajectory (and not just 20-step tours). We presented results (both quantitative as well as qualitative) for this full-scale generalization in the main paper. In this section, for completeness, we also provide results of all our models and baselines on the 20-step "short tours" in Tab. 1. We also show qualitative examples of predictions made by our agent for these 20-step short tours in Fig. 5. All trends observed and discussed in the main paper (Section 6) hold true for Tab. 1 as well.

We'd like to reiterate (from Section 3 in the main paper) that generating and using these "short tours" merely happens during train. The focus of our task (as well as the subject of all our evaluations/analysis in the main paper) is on the full-scale exploration tours and maps for entire scenes.

4. Additional Qualitative EMQA Results

We show additional qualitative results on both "short" (Fig. 5) and "full" (Fig. 6) tours for our proposed EMQA model.

5. Sim2Real: Real-world RGBD Results

Recall that Fig. 4 (c) in the main paper had qualitative results of evaluating our model on a more challenging, real-world RGB-D dataset [10] with imperfect depth+pose and camera jitter. Here, we provide some additional details about the same.

We perform zero-shot evaluation tests on this dataset – we construct scene memories using our pre-trained SMNet [2] model, hand-craft questions relevant to the scene and generate prediction output using our pre-trained LingUNet-based question-answering module (no component of our model was fine-tuned on [10]). We pre-processed the input video by subsampling the original frame sequence (selecting every 10th frame). We discarded visually blurry images by looking at the variance of the Laplcaian in the RGB image. Any image with a variance below a threshold of 100 is discarded. Data inputs are grouped into tuples of (RGB, Depth and Pose) using timestamps. We allow for a max timestamp difference of at most 0.02s between modalities. The depth frame is further pre-processed through a binary erosion step with a circular element of radius 20 pixels.

In the answer to the question 'where did you first see the chair?' (Fig. 4 (c) in the main paper), we see the first two chairs (at the bottom) of the map being detected. The third chair on the other side of the desk is also detected. Although it is not, strictly speaking, the first instance of a chair seen in the video, that chair appears at the beginning of the video. In the answer to the question 'where did you last see the chair', that third chair at the top of the map is not detected this time. The bottom two chairs are detected again because there are also seen last in the video. In the answer to the question 'where did you see the chair', three

Method	Top-down map output space			Egocentric pixel output space		
	IoU	Recall	Precision	IoU	Recall	Precision
LangOnly	15.09 ± 0.22	22.36 ± 0.26	29.56 ± 0.41	15.23 ± 0.23	23.19 ± 0.27	29.61 ± 0.43
EgoSemSeg	27.51 ± 0.32	39.24 ± 0.48	**55.53** ± 0.58	29.05 ± 0.36	39.45 ± 0.50	**59.57** ± 0.55
SMNetDecoder	30.76 ± 0.37	45.06 ± 0.46	54.51 ± 0.37	31.36 ± 0.31	45.86 ± 0.23	55.52 ± 0.54
EgoBuffer-Avg [4]	2.46 ± 0.09	4.53 ± 0.17	8.10 ± 0.26	2.18 ± 0.09	4.29 ± 0.18	7.98 ± 0.28
EgoBuffer-GRU [1]	1.39 ± 0.06	2.02 ± 0.11	9.09 ± 0.22	1.34 ± 0.07	2.08 ± 0.12	9.36 ± 0.23
EgoBuffer-Attn [5]	2.18 ± 0.07	3.67 ± 0.12	10.08 ± 0.24	2.01 ± 0.07	3.66 ± 0.12	10.15 ± 0.26
Ours	36.09 ± 0.53	48.03 ± 0.62	53.04 ± 0.51	36.66 ± 0.27	49.64 ± 0.41	53.12 ± 0.32
Ours (+temporal)	**37.72** ± 0.45	**49.88** ± 0.55	53.48 ± 0.64	**38.29** ± 0.44	**51.47** ± 0.58	53.48 ± 0.59

Table 1. EMQA results on "short tours" for our proposed model and baselines in the "top-down map" and "egocentric pixel" output space.

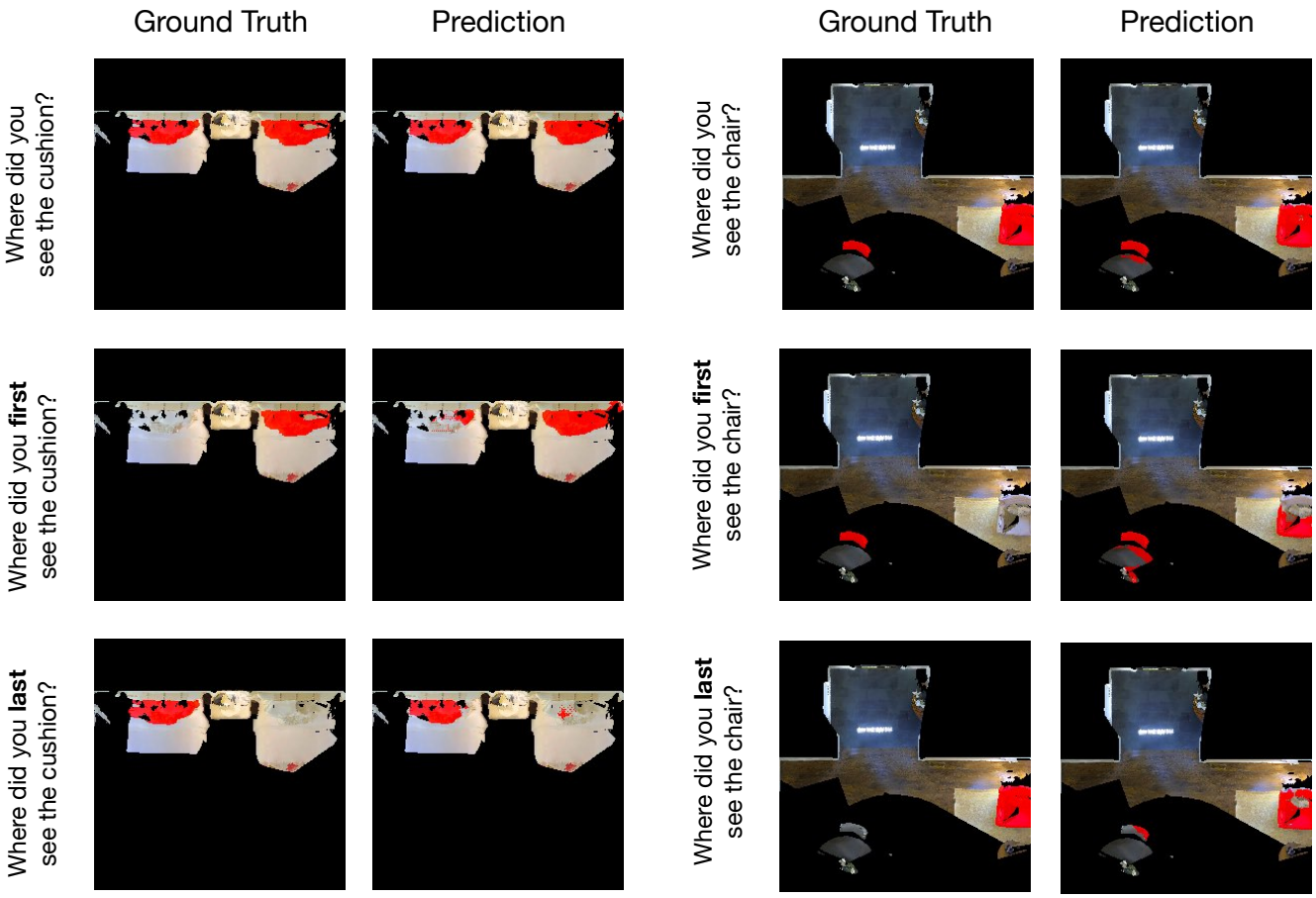

Figure 5. We provide qualitative results of our proposed EMQA agents grounding answers to questions onto the top-down environment floorplan for "short" tours.

chairs are detected. The fourth one with the stuffed animal on it has been missed.

6. Sim2Real: Noisy Pose Experiments

In this section, we discuss the impact of localization noise on the construction of scene memory representations and downstream question answering performance.

6.1. Noise Models

As discussed in Sec. 6 of the main paper (subsec: Sim2Real Robustness), we investigate the impact of two qualitatively different types of noise – (1) noise sampled from a real-world robot [9], **independently** added to the pose at each step along the ground truth trajectory, and (2) **cumulatively** integrating per-step noisy pose change esti-

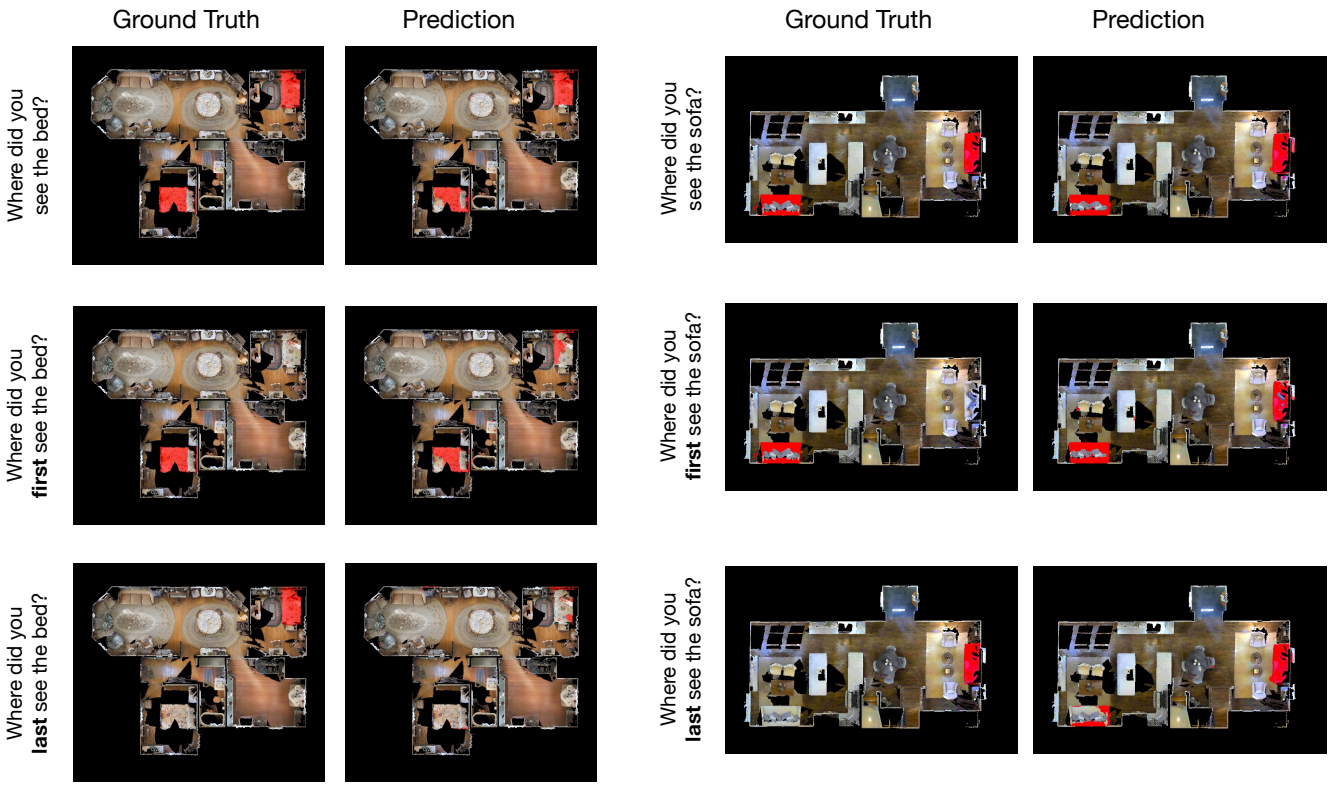

Figure 6. We provide additional qualitative results of our proposed EMQA agents grounding answers to questions onto the top-down environment floorplan for "full" tours.

mates derived from a visual odometry-based egomotion estimation module.

In this section, we discuss how we implement and integrate these two noise models into the pose at each step of our guided exploration tours.

Independent Noise. Here, we leverage the actuation noise models derived from the real-world benchmarking [7] of a physical LoCoBot [9]. Specifically, this comprises the linear and rotational actuation noise models. The linear noise distribution is a bi-variate Gaussian (with a diagonal covariance matrix) that models localization inaccuracies along the X (orthogonal to the heading on the ground plane) and Z axis (direction of heading on the ground plane), whereas the rotational noise distribution is a uni-variate Gaussian for the agent's heading. We integrate these noise models in the guided tours by independently adding noise to each ground truth pose of the assistant along the tour. Additionally, we implement this setup with two levels of noise multipliers – 0.5x and 1x to simulate varying intensities of noise being added. An example of a trajectory with this noise model has been visualized in Fig. 5(a) in the main paper.

Cumulative Noise. Beyond adding noise to existing ground truth trajectory pose, we additionally investigate a setting where our model is given an initial pose and from thereon, it estimates per-step changes in its pose solely from observations (visual odometry) and integrates these predictions along the trajectory, thereby maintaining an up-to-date (though noisy) estimate of its current pose

In contrast to the independent addition of noise to the ground truth pose at each tour step, this is a more challenging setting where the assistant maintains and works with a pose estimate completely on its own. As a result, noise picked up at each step (e.g. during collisions) cascades and accumulates throughout the rest of the trajectory. We present an example of a noisy egomotion trajectory in Fig. 5(b). We now describe the visual odometry model that we adapt from [12] and use to estimate per-step pose changes.

6.2. Visual Odometry Models

Model Architecture. The visual odometry (VO) model takes as input a pair of consecutive RGB-D observations $(o_t,\ o_{t+1})$ and estimates three relative pose transformation components – the translation along X and Z axis, and the rotation angle (θ). For our experiments, we employ a stripped-down (bare-bones) version of the architecture proposed in [12]. The architecture includes a Resnet-18 [6] feature extractor backbone followed by two fully-connected

(FC) layers with Dropout. We do away with the soft top-down projection and discretized depth inputs, and only use RGB+Depth observation pairs from successive steps. Additionally, as suggested by the authors [12], we train action-specific models to deal with the differences in the action distributions.

Dataset Preparation. The pre-trained VO models provided by the authors of [12] were trained on the Gibson dataset of 3D scans [11]. In contrast, we work with scenes from the Matterport3D [3] dataset. More importantly, these models were trained in a setup that used agent actuation specifications (forward step: 25cm, rotation angle: 30°) that are significantly different from those used in our exploration tours (forward step: 10cm, rotation angle: 9°). Therefore, a zero-shot transfer of models pre-trained by the authors in [12] to our setup is likely to not work well. We qualitatively verify this in Fig. 7. Note that the trajectory formed by integrating predictions from a pre-trained VO model applied directly to our setup doesn't match the original trajectory at all.

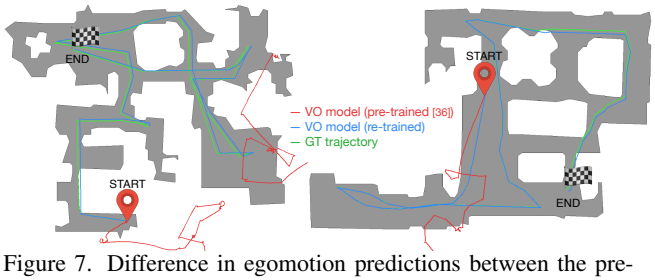

Figure 7. Difference in egomotion predictions between the pre-trained VO model provided by the authors of [12] and the model we retrained on a custom dataset of 100k data points from Matterport3D [3] scenes. Apart from the difference in the scene datasets, the provided pre-trained models were trained under significantly dissimilar actuation specifications. This discrepancy called for a re-training of the VO model under the guided tour setup we use [2].

To remedy this, we retrain the VO models from [12] on scenes from the Matterport3D dataset and under the actuation specificns of EMQA. To do that, we first create a separate VO dataset in Matterport3D [3]. For generating the dataset, we follow the protocol laid out in [12]. We sample (uniformly, at random) two navigable points in the scene and compute the shortest navigable path between the two points. Then, we sample (again, uniformly, ar random) consecutive observation pairs along these shortest-path trajectories. Using this method, we create a dataset of 100k and 20k training and validation data points.

Training and Evaluation. Following the training regime proposed by [12], we train our model by jointly minimizing the regression and geometric invariance losses from [12]. Fig. 7 (blue) shows the trajectory generated via integration of predictions from the retrained VO model. We note a significant qualitative improvement in the trajectory as it much closely resembles the ground truth trajectory. In addition to that, in Tab. 2, we also quantitatively analyze the improvements gained through retraining by comparing the average deviation in the trajectories (as measured by the average over per-step pose RMSEs) predicted via the pre-trained and the re-trained VO models. We note that with a retrained VO model, we significantly improve the quality of our trajectories by getting $\sim$ 6x less deviation with the ground truth trajectory.

For completeness, Tab. 2 also contains metrics for trajectories generated by independently adding noise to the ground-truth pose. It is evident that both (a) increasing the intensity of independently added noise and (b) moving from independent to VO-based cumulative noise leads to larger deviations in the predicted trajectories from the ground truth ($\sim$1.3x and $\sim$15x increase respectively).

Method	RMSE (X axis)	RMSE (Z axis)
Noise 0.5x (independent)	0.068	0.066
Noise 1x (independent)	0.09	0.088
Egomotion (re-trained)	1.448	1.346
Egomotion (pre-trained [12])	11.074	10.386

Table 2. Average absolute and relative differences between the ground truth poses and their corresponding noisy estimates at each step. Here, we highlight the improvement in egomotion estimation by retraining the VO model for our setup. We also note how, due to its cumulative nature (accumulating noise along the trajectory), the error metrics are much higher for the egomotion predictions compared to the independent noise models (where noise is independently added to the pose at each step).

6.3. Evaluating EMQA Models on Noisy Pose Data

In the discussions so far, we talked about adding noise (of various types and intensities) to the pose information in our dataset. In this section, we first investigate the performance of our models under the noisy settings and then, make a case for re-training our EMQA models so that they learn to adapt to noisy pose inputs.

Towards that end, we plot the improvement in performance upon retraining across three setups in Fig. 8. In each of the sub-figures, we first plot the original model's performance (trained and evaluated using ground truth pose i.e. train=GT, eval=GT). This is followed by taking the same model (trained using ground truth pose) and evaluating it under the noisy pose setting (i.e. train=GT, eval=Noise1x). And finally, we re-train our model so that it adapts to the noise and then evaluate this re-trained model in the noisy setting (train=Noise1x, eval=Noise1x).

We now compare the above settings across three experiments.

1. Semantic map prediction performance of SMNet (Fig. 8 (a)): Our method utilizes scene memory representations learnt by SMNet for downstream question answering. As

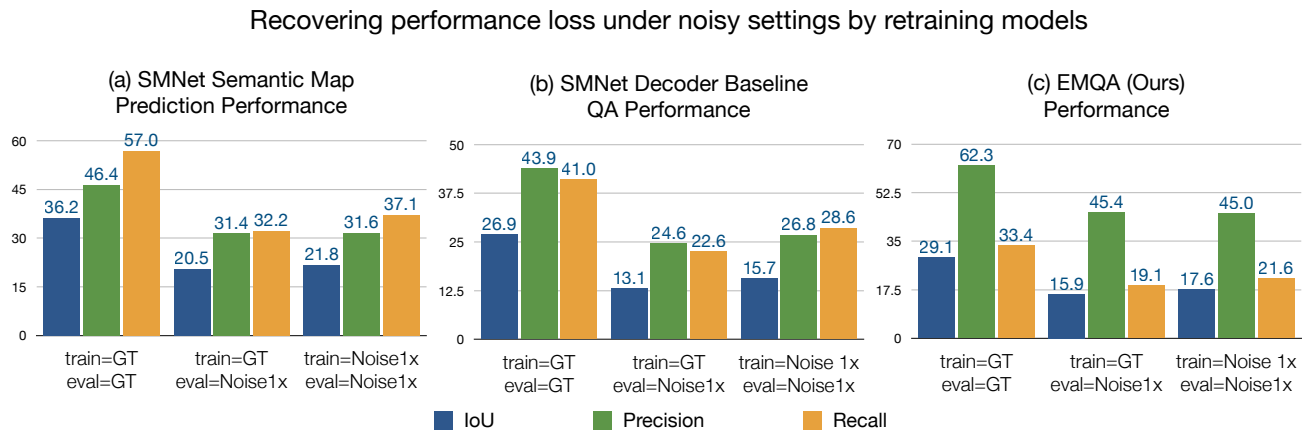

Figure 8. Recovering performance loss under noisy settings in semantic map prediction (a) and question answering (b,c) by retraining SMNet and LingUNet on noisy pose data. In each sub-figure, we observe a dip in the evaluation metrics of the ground truth (GT) pose-trained model upon adding noise to the input pose. This drop is partially recovered from, by retraining the model(s) on noisy data.

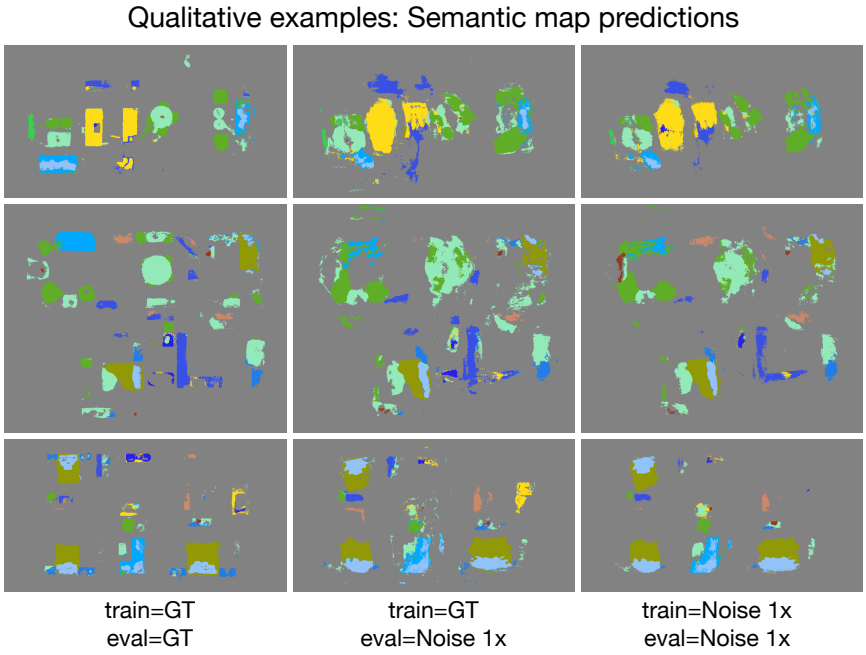

Figure 9. We show qualitative examples of semantic map predictions for the three experiments described in Sec. 6.3 – (1) our model trained and evaluated using ground truth pose (left), (2) our model trained using ground-truth pose, but evaluated in the noisy pose setting (middle) and (3) our model retrained using noisy pose (right). We see sharper boundaries and less label splatter upon retraining with noise.

a result, comparing the semantic map predictions serves as a proxy for the quality of scene representations. Here, we observe a 43% drop in IoU on adding noise and evaluating using the original model. Upon retraining SMNet with noisy data, we see a 6% gain on the same metric.

2. Question answering (QA) performance of SMNet Decoder baseline [2] (Fig. 8 (b)): Here, we plot the downstream question-answering performance for the SMNetDecoder baseline (our best performing baseline from Sec. 5 in the main paper). We observe a 41% drop in IoU on adding noise and evaluating using the original model. However, we see a 19% increase on the same metric when we retrain SMNet on noisy data.

3. Question answering (QA) performance of our method (Fig. 8 (c)): Finally, we plot the downstream question-answering performance for our proposed method. Initially, on integrating noise and evaluating using the original model, we observe a 39% drop in IoU performance. How-

ever, upon retraining SMNet and LingUNet, we see a 10% increase on the same metric.

Across all the three experiments, we observe that (a) when models trained using privileged, oracle pose information are evaluated with noise in pose, their performance (understandably) drops and (b) we are able to recover the drop by re-training so that the model learns to adapt to its noisy inputs. We also qualitatively demonstrate the improvement in semantic map predictions obtained through a re-training of our models to adapt to noisy pose in Fig. 9 (note the sharper boundaries and reduced label splatter).

References

[1] Bram Bakker. Reinforcement learning with long short-term memory. In *NIPS*, pages 1475–1482, 2001. 5

[2] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semantic maps and representations from egocentric views, 2017. 1, 4, 7, 8

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 1, 7

[4] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. 5

[5] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 538–547, 2019. 5

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 6

[7] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020. 6

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 3

[9] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019. 5, 6

[10] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012. 4

[11] Fei Xia, Amir Roshan Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. *CoRR*, abs/1808.10654, 2018. 7

[12] Xiaoming Zhao, Harsh Agrawal, Dhruv Batra, and Alexander G Schwing. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16127–16136, 2021. 6, 7