EGOCENTRIC SPATIAL MEMORY NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatial memory, or the ability to remember and recall specific locations and objects, is central to autonomous agents' ability to carry out tasks in real environments. A key challenge here is dealing with partial observability (PO) as cameras and other sensors can only observe the parts of the world that are visible from their specific vantage point. The most common approach for dealing with PO is to aggregate observations in an external memory. However, most existing external memory modules do not leverage any spatial inductive bias. In this paper, we propose Egocentric Spatial Memory Networks (ESMN), which encodes the memory in an ego-sphere around the agent, enabling expressive 3D representations. ESMN can be trained via either imitation or reinforcement learning and improves both training efficiency and final performance against other memory baselines on both drone and manipulator visuomotor control tasks. The explicit ego-centric geometry also enables us to seamlessly combine the learned controller with other non-learned modalities, such as local obstacle avoidance. We further show applications to semantic segmentation on the ScanNet dataset, where ESMN naturally combines image-level and map-level inference modalities. Through our broad set of experiments, we show that ESMN provides a general computation graph for embodied spatial reasoning, and the module forms a bridge between real-time mapping systems and differentiable memory architectures.

1 INTRODUCTION

Egocentric spatial memory is central to our understanding of spatial reasoning in biology (Klatzky, 1998; Burgess, 2006), where an embodied agent constantly carries with it a local map of its surrounding geometry. Such representations have particular significance for action selection and motor control (Wang et al., 2018; Hinman et al., 2019). For robotics and embodied AI, the benefits of a persistent local spatial memory are also clear. Such a system has the potential to run indefinitely, and bypass both the memory and runtime complexities of large scale world-centric mapping. Some pieces of work in the mapping literature have already utilized this ego-centric formulation, with applications to drone obstacle avoidance (Fragoso et al., 2018), mono-to-depth from video (Liu et al., 2019), and planar robot mapping (Fankhauser et al., 2014).

In parallel with these explicit ego-centric mapping systems, a new paradigm of differentiable memory architectures has arisen, where an explicit memory bank is augmented to a neural network which can then learn read and write operations (Weston et al., 2014; Graves et al., 2014; Sukhbaatar et al., 2015). When compared to Recurrent Neural Networks (RNNs), the persistent memory circumvents issues of vanishing or exploding gradients, enabling solutions to complex long-horizon tasks. These have also been successfully applied to visuomotor control and navigation tasks (Wayne et al., 2018), surpassing baselines such as the ubiquitous Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997).

We focus on the intersection of these two branches of research, and propose Egocentric Spatial Memory Networks (ESMN), a parameter-free module that encodes the memory in an ego-sphere around the agent. We emphasize the suitability of an ego-centric mapping computation graph for propagating gradients over long time horizons in an end-to-end manner, enabling an ego-centric memory of learned features, see Figure 1. We demonstrate the superiority of learning features through the ESMN module, on both drone target shape reaching and object segmentation tasks.

For other visuomotor control tasks, we show that even without learning features through the module, ESMN is able to consistently outperform baselines approaches. Through these experiments, we show

that the applications of our module are widespread, where it can either be dropped into existing pipelines as a non-learned module, or end-to-end trained in a larger computation graph, depending on the task requirements.

2 Related Work

2.1 Mapping

Geometric mapping is a mature field, with many solutions available for constructing high quality maps. Such systems typically maintain an allocentric map, either by projecting points into a global world co-ordinate system (Newcombe et al., 2011a; Whelan et al., 2015), or by maintaining a certain number of keyframes in the trajectory history (Newcombe et al., 2011b; Zhou et al., 2018; Bloesch et al., 2018; Clark et al., 2018). If these systems are to be applied to lifelong embodied AI, then strategies must then be adopted in order to effectively select the parts of the map which are immediately useful, and discard the rest from memory (Cadena et al., 2016).

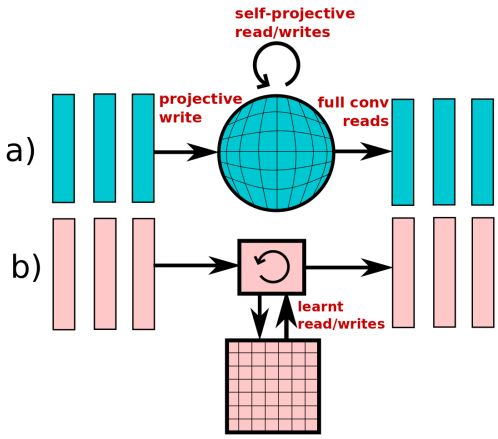

Figure 1: Computation graphs of (a) ESMN and (b) neural memory (MemNN, NTM)

For robotics applications, prioritizing geometry in the immediate vicinity is a sensible prior. Rather than taking a world-view to map construction, such systems often formulate the mapping problem in a purely ego-centric manner, performing continual re-projection to the newest frame and pose with fixed-sized storage. Cigla et al. (2017) use per-pixel depth Gaussian Mixture Models (GMMs) to maintain an ego-cylinder of belief around a drone, with applications to collision avoidance (Fragoso et al., 2018). Fankhauser et al. (2014) use ego-centric height maps, and demonstrate on a quadrupedal robot walking over obstacles. In a different application, Liu et al. (2019) learn to predict depth images from a sequence of RGB images, again using ego reprojections. These systems are designed to represent only at the level of single-frame depth and RGB features. For mapping more expressive implicit features via end-to-end training, a fully differentiable long-horizon computation graph is required. Any computation graph which satisfies this requirement is generally referred to as memory in the neural network literature.

2.2 Memory

The concept of memory in neural networks is deeply coupled with recurrence. Naive recurrent networks have vanishing and exploding gradient problems (Hochreiter, 1998), which LSTMs (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho et al., 2014) mediate using additive gated structures. More recently, dedicated differentiable memory blocks have become a popular alternative. Weston et al. (2014) applied Memory Networks (MemNN) to question answering, using hard read-writes and separate training of components. Graves et al. (2014) and Sukhbaatar et al. (2015) instead made the read and writes 'soft' with the proposal of Neural Turing Machines (NTM) and End-to-End Memory Networks (MemN2N) respectively, enabling joint training with the controller. Other works have since conditioned dynamic memory on images, for tasks such as visual question answering (Xiong et al., 2016) and object segmentation (Oh et al., 2019). Another distinct but closely related approach is self attention (Vaswani et al., 2017). These approaches also use key-based content retrieval, but do so on a history of previous observations with adjacent connectivity. Despite the lack of geometric inductive bias, recent results demonstrate the amenability of general memory (Wayne et al., 2018) and attention (Parisotto et al., 2019) to visuomotor control and navigation tasks.

Other authors have explored the intersection of network memory and spatial mapping for navigation, but have generally been limited to 2D aerial-view maps, focusing on planar navigation tasks. Gupta et al. (2017) used an implicit ego-centric memory which was updated with warping and confidence maps for discrete action navigation problems. Parisotto & Salakhutdinov (2017) proposed a similar

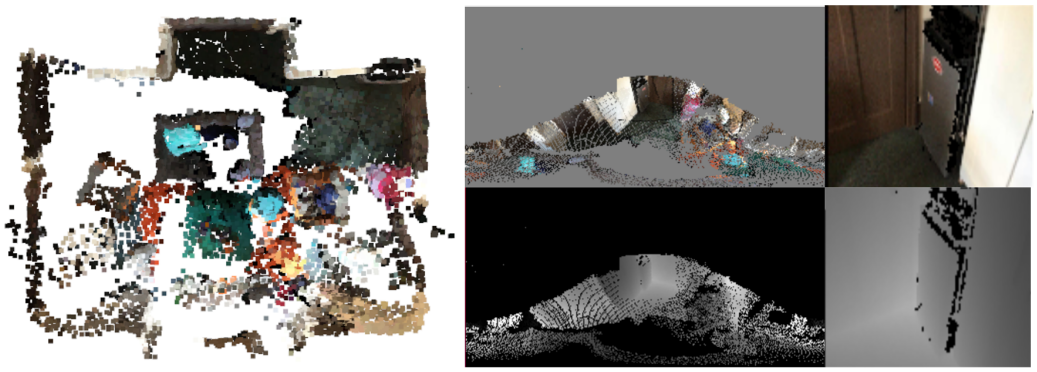

Figure 2: Left: point cloud representation of the ego-centric memory around the camera after full rotation in a scene from ScanNet, with RGB features. Mid: (top) Equivalent omni-directional RGB image, (bottom) equivalent omni-directional depth image, both without smoothing to better demonstrate the quantization holes. Right: (top) A single RGB frame, (bottom) a single depth frame

setup, but used dedicated learned read and write operations for updates, and tested on simulated Doom environments. Without consideration for action selection, Henriques & Vedaldi (2018) proposed a similar system, but instead used an allocentric formulation, and tested on free-form trajectories of real images. Our memory is much more expressive than these 2D examples, with the ability to represent detailed 3D geometry in all directions around the agent.

3 METHOD

In this section, we describe the formulation of the memory module. The module operates as an Extended Kalman Filter (EKF), with an ego-sphere image $\mu_t \in \mathbb{R}^{h_s \times w_s \times (1+n)}$ and its diagonal covariance $\Sigma_t \in \mathbb{R}^{h_s \times w_s \times (1+n)}$ representing the state. The ego-centric state image consists of 1 channel for radial depth and n for encoded features. Image measurements are assumed to be captured from projective depth cameras, which similarly store 1 channel for depth and n for encoded features. We also assume incremental pose measurements $u_t \in \mathbb{R}^6$ with covariance $\Sigma_{u_t} \in \mathbb{R}^{6\times 6}$ are available, in the form of a translation and rotation vector. The algorithm overview is presented in Algorithm 1.

First, the motion step takes the state from the previous frame μ_{t-1}, Σ_{t-1} and transforms this into a predicted state for the current frame $\bar{\mu}_t, \bar{\Sigma}_t$ via functions f_m, F_m and the control vector u_t with covariance Σ_{u_t}. Then in the observation step, we use measured visual features $(v_{ti} \in \mathbb{R}^{h_{vi} \times w_{vi} \times (1+n)})_{i \in \{1,\dots,m\}}$ with diagonal covariances $(V_{ti} \in \mathbb{R}^{h_{vi} \times w_{vi} \times (1+n)})_{i \in \{1,\dots,m\}}$, and pose measurements $(p_{ti} \in \mathbb{R}^6)_{i \in \{1,\dots,m\}}$ with covariances $(P_{ti} \in \mathbb{R}^{6\times 6})_{i \in \{1,\dots,m\}}$, to produce a new observation of the state $\hat{\mu}_t \in \mathbb{R}^{h_s \times w_s \times (1+n)}$, again with diagonal covariance $\hat{\Sigma}_t \in \mathbb{R}^{h_s \times w_s \times (1+n)}$, via functions f_o and F_o. The measured poses also take the form of a translation and rotation vector.

Finally, the update step takes our state prediction $\bar{\mu}_t, \bar{\Sigma}_t$ and state observation $\hat{\mu}_t, \hat{\Sigma}_t$, and fuses them to produce our new state belief μ_t, Σ_t. We spend the remainder of this section explaining the form of the constituent functions. All functions in Algorithm 1 involve re-projections across different image frames, using forward warping. Functions f_m, F_m, f_o and F_o are therefore all built using the same core functions. While the re-projections could be solved using a typical rendering pipeline of mesh construction followed by rasterization, we instead choose a simpler approach and directly quantize the pixel projections with variance-based image smoothing to fill in quantization holes. An overview of the projection and quantization steps for the mean belief image is shown in Fig. 3.

3.1 FORWARD WARPING

Forward warping projects ordered equidistant pixel co-ordinates from frame f_1 to non-ordered non-equidistant pixel co-ordinates in frame f_2. We use $\tilde{\mu}_2 = \{\tilde{d}_2, \tilde{e}_2\}$ to denote the loss of ordering following projection from $\mu_1 = \{d_1, e_1\}$, where d and e represent depth and encoded features

Algorithm 1: Ego-sphere EKF

1 Given: f_m, F_m, f_o, F_o
2 $\bar{\mu}_t = f_m(u_t, \mu_{t-1})$
3 $\bar{\Sigma}_t = F_m(u_t, \mu_{t-1}, \Sigma_{t-1}, \Sigma_{u_t})$
4 $\hat{\mu}_t = f_o((v_{ti}, p_{ti})_{i \in I})$
5 $\hat{\Sigma}_t = F_o((v_{ti}, p_{ti}, V_{ti}, P_{ti})_{i \in I})$
6 $K_t = \bar{\Sigma}_t \left[\bar{\Sigma}_t + \hat{\Sigma}_t\right]^{-1}$
7 $\mu_t = \bar{\mu}_t + K_t[\hat{\mu}_t - \bar{\mu}_t]$
8 $\Sigma_t = [I - K_t]\,\bar{\Sigma}_t$
9 return μ_t, Σ_t

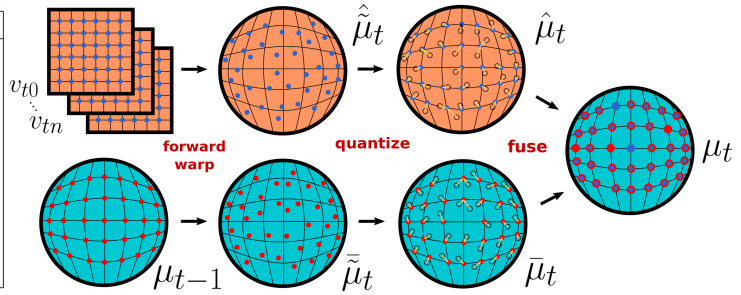

Figure 3: Algorithm overview, and visualization of the projection and quantization steps, used to bring the belief from the previous agent frame to the current agent frame.

respectively. We only consider warping from projective to omni cameras, which corresponds to functions f_o, F_o, but the omni-to-omni case as in f_m, F_m is identical except with the inclusion of another polar co-ordinate transformation.

The encoded features are assumed constant during projection $\tilde{e}_{f2} = e_{f1}$. For depth, we must transform the values to the new frame in polar co-ordinates, which is a composition of a linear transformation and non-linear polar conversion. The full projection is composed of a scalar multiplication with homogeneous pixel co-ordinates p_{f1}, transformation by camera inverse intrinsics K_1^{-1} and frame-to-frame T_{12} matrices, and polar conversion f_p:

$$\{\tilde{\phi}_{f2}, \tilde{\theta}_{f2}, \tilde{d}_{f2}\} = f_p(T_{12} K_1^{-1}[p_{f1} \odot d_{f1}]) \tag{1}$$

Combined, this provides us with both the forward warped image $\tilde{\mu}_{f2} = \{\tilde{d}_{f2}, \tilde{e}_{f2}\}$, and the newly projected polar pixel co-ordinates $\tilde{pp}_{f2} = \{k_{ppr}\tilde{\phi}_{f2}, k_{ppr}\tilde{\theta}_{f2}\}$, where k_{ppr} denotes the pixels-per-radian resolution constant. The variances are also projected using the full analytic Jacobians, which are efficiently implemented as tensor operations, avoiding costly autograd Jacobian computation.

$$\hat{\tilde{\Sigma}}_2 = J_V V_1 J_V^T + J_P P_{12} J_P^T \tag{2}$$

3.2 Quantization, Fusion and Smoothing

Following projection, the quantization operation first rounds the polar floating point pixel projections $\tilde{pp}_{f2}$ into integer pixel co-ordinates pp_{f2}. This in general leads to quantization holes and duplicates. The duplicates are handled with a variance conditioned depth buffer, such that the closest projected depth is used, provided that it's variance is lower than a set threshold. This in general prevents highly uncertain close depth values from overwriting highly certain far values. We then perform per pixel fusion based on lines 6 and 7 in Algorithm 1. Finally, we perform variance based image smoothing, whereby we treat each $N \times N$ image patch $(\mu_{k,l})_{k \in \{1,..,N\}, l \in \{1,..,N\}}$ as a collection of independent measurements of the central pixel, and combine their variance values based on central limit theory, resulting in smoothed values for each pixel in the image $\mu_{i,j}$. Although we use this to update the belief, we do not smooth the variance values, meaning projection holes remain at prior variance. This prevents the smoothing from distorting our belief during subsequent projections, and makes the smoothing inherently local to the current frame only. The smoothing formula is as follows:

$$\mu_{i,j} = \frac{\sum_k \sum_l \mu_{k,l} \cdot \sigma_{k,l}^2}{\sum_k \sum_l \sigma_{k,l}^2} \tag{3}$$

Given that the quantization is a discrete operation, we cannot compute it's analytic jacobian for uncertainty propagation. We therefore approximate the added quantization uncertainty using the numerical pixel gradients of the newly smoothed image $G_{i,j}$, and assume additive noise proportional to the x and y quantization distances $\Delta pp_{i,j}$:

$$\Sigma_{i,j} = \tilde{\Sigma}_{i,j} + G_{i,j} \Delta pp_{i,j} \tag{4}$$

4 EXPERIMENTS

In this section we evaluate ESMN on a range of tasks including a range of image-to-action learning tasks and object segmentation.

4.1 IMAGE-TO-ACTION LEARNING

We first test the utility of our memory module on two simulated visual reacher tasks, which we refer to as Drone Reacher (*DR*) and Manipulator Reacher (*MR*). Both are implemented using the CoppeliaSim robot simulator (Rohmer et al., 2013), and its Python extension PyRep (James et al., 2019). We implement DR ourselves, while MR is a modification of the reacher task in RLBench (James et al., 2020), with a FRANKA EMIKA Panda robot. Both tasks consist of 3 targets placed randomly in a simulated arena, and colors are newly randomized for each episode. For DR, the targets consist of a cylinder, sphere, and "star", and for MR the targets are all spheres, see Figure 4.

In both tasks, the target locations remain fixed for the duration of an episode, and the agent must continually navigate to newly specified targets, reaching as many as possible in a fixed time frame of 100 steps. The targets are specified to the agent either as RGB color values or shape class id, depending on the experiment. The agent does not know in advance which target will next be specified, meaning a memory of all targets and their location in the scene must be maintained for the full duration of an episode. Both environments have a single body-fixed camera, as shown in Figure 4, and also multiple external scene cameras, which we use separately for different experiments.

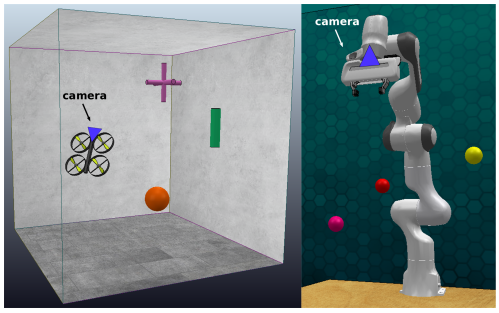

Figure 4: Visualization of (a) Drone Reacher and (b) Manipulator Reacher tasks.

Action spaces of joint velocities $\dot{q} \in \mathbb{R}^7$ and cartesian velocities $\dot{x} \in \mathbb{R}^6$ are used for *MR* and *DR* respectively. Expert translations move the end-effector or drone directly towards the target, and expert rotations rotate the camera towards the target via shortest rotation. Expert joint velocities are calculated for linear end-effector motion via the Jacobian. For all experiments, we compare to baselines of single-frame, dual-stacked LSTM, and NTM. Full details of the training setups are provided in Appendix A.1. The results for all experiments are presented in Table 1.

	DR Ego S	DR Ego C	DR SF C	DR SM C	MR Ego C	MR SF C	MR SM C
Mono	1.9(2.0)	2.1(2.2)	7.8(9.0)	8.4(9.5)	1.6(1.4)	0.0(0.2)	0.0(0.2)
LSTM 256	10.3(3.1)	10.7(3.4)	3.7(5.5)	3.4(3.6)	1.1(0.9)	0.0(0.2)	0.1(0.3)
LSTM 1024	11.3(3.6)	11.6(3.6)	4.3(4.3)	2.8(1.9)	1.0(0.7)	0.0(0.2)	0.1(0.3)
NTM	12.3(1.9)	12.0(3.9)	3.4(3.2)	3.4(3.2)	0.9(0.5)	0.0(0.2)	0.1(0.2)
ESM	4.8(3.8)	**20.6**(9.8)	**16.7**(12.9)	**15.8**(12.9)	**10.2**(4.3)	**7.6**(6.6)	**7.4**(4.4)
ESMN	**18.0**(6.8)	**20.4**(9.1)	**16.5**(13.2)	**16.3**(12.2)	**10.9**(4.0)	**7.8**(5.9)	**7.5**(6.4)
Random	0.2(0.4)	0.2(0.4)	0.2(0.4)	0.2(0.4)	0.6(1.5)	0.6(1.5)	0.6(1.5)
PO	5.2(4.5)	21.5(9.6)	19.0(12.0)	18.7(11.9)	11.4(3.6)	8.5(6.0)	8.2(5.5)
Expert	22.4(9.9)	22.4(9.9)	22.4(9.9)	22.4(9.9)	14.7(5.3)	14.7(5.3)	14.7(5.3)

Table 1: Final policy performances on the various drone reacher (DR) and manipulator reacher (MR) tasks, from ego-centric (Ego), scene-fixed (SF) and scene moving (SM) cameras, conditioned on either target color (C) or shape (S). The values indicate the mean number of targets reached in the 100 time-step episode, and the variance, when averaged over 256 runs. Environment seeds remain fixed between experiments.

4.1.1 FROM EGO-CENTRIC OBSERVATIONS

Our first set of experiments take observations from body-mounted cameras. We can see in Table 1 that for both DR and MR, our module significantly outperforms other memory baselines, which do not explicitly incorporate geometric inductive bias. Clearly, the baselines have difficulty in optimally interpreting the stream of incremental pose measurements and depth, and are unable to remember all the observed regions in the agent ego-sphere. In contrast, ESM by design stores features in the memory with meaningful indexing. The inclusion of relative cartesian co-ordinates in the memory image also effectively aligns each pixel with an associated relative translation. When fed to the network post-ESM, action selection can then in principle be simplified to feature matching, reading the associated relative co-ordinates, and then transforming to the required action space. A collection of short memory sequences for the various tasks are presented in Figure 5, with (a), (b) and (d) coming from ego-observations. In all cases we see the agent reach one target by the third frame, before immediately re-orienting to reach the next target.

We also observe that gradientless ESM performs well when conditioned on color, but fails catastrophically when conditioned on shape id. This is to be expected, as the ability to retrieve shape information from the memory is strongly influenced by the ESM resolution, the quantization holes, and the strong angular distortion. However, when training end-to-end with a convolutional encoder, we can see that these issues are alleviated, and ESMN learns to store useful mono-extracted features in memory, enabling selection of the correct expert actions when later required. We show the activations of the 3 most dominate feature channels projected to RGB space during a short sequence in Figure 5 (a).

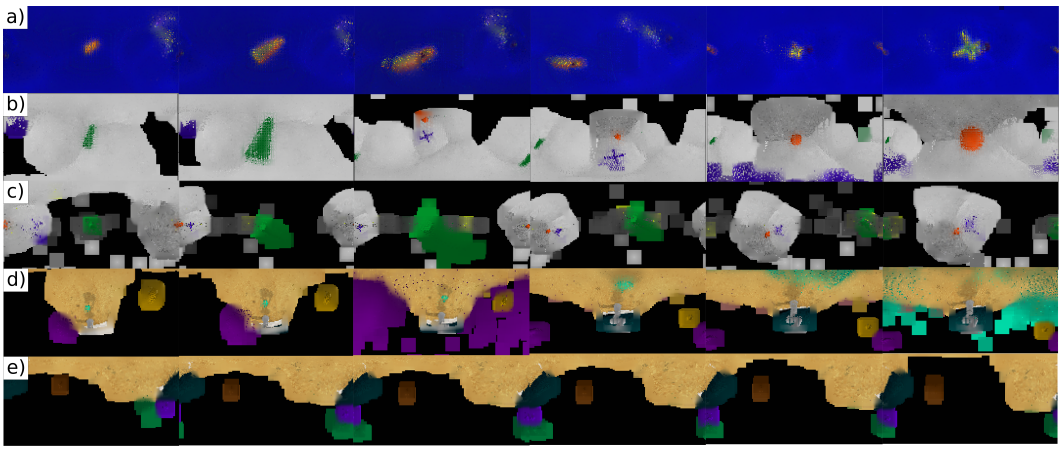

Figure 5: Sample trajectories through the memory for (a) ESMN on DR-Ego-S, and ESM on (b) DR-Ego-C, (c) DR-SF-C, (d) MR-Ego-C, (e) MR-SF-C

4.1.2 FROM SCENE-CENTRIC OBSERVATIONS

Our next set of experiments involve images from cameras external to the agent. The poses of these cameras are randomized for each episode during training, and follow random motions, with a strong bias to face towards the centre of the scene. For fairer comparison against the baselines, we also condition these networks on the absolute pose of the scene camera, rather than providing only incremental poses as used in ESM. Here we explore the ability of ESM to generalize both to unseen camera poses and to camera motion. Again, we see that the baselines fail to learn meaningful spatial representations. In contrast, the strong inductive bias in ESM enables the task to be solved, whether the camera is static or moving. The memories in these tasks take on a different profile, as can be seen in Figure 5 (c) and (e). While ego-observations always contain information in the centre, where the most recent frame projects with high density, this is not the case for arbitrarily positioned cameras. Sequence (e) actually corresponds to the brown target being reached, but is only represented by two pixels in the memory due to the distance of the scene camera, and so we do not observe the object expansion as seen in other sequences. The large apparent area is a result of the variance-based smoothing, for a high confidence pixel surrounded by low confidence ones (holes).

	O	OA	E2S	S2E
ESM-DepthAvoid	-	32.1(40.7)	-	-
ESM	68.4(42.9)	23.8(33.0)	6.8(6.5)	11.1(6.2)

Table 2: Number of collisions during an episode for obstacle avoidance task, with obstacles but no avoidance (O), obstacles with avoidance (OA), for both depth based avoidance (ESM-DepthAvoid) and ESM based avoidance (ESM), and the number of targets reached in the standard obstacle-less environment for transfer learning ego-to-scene (E2S) and scene-to-ego (S2E) policies for ESM on the MR task.

4.1.3 Reinforcement Learning

While our imitation learning experiments demonstrate the superior ability of ESM to generalize from offline datasets to successful policies, the formulation of assuming expert actions in PO environments is inherently limited. It is not necessarily true that the best action always rotates the camera directly to the next target for example. In general, for finding optimal policies in PO environments, methods such as reinforcement learning (RL) must be used. We therefore train both ESM and an LSTM baseline on a similar sequential target reacher task via DQN (Mnih et al., 2015), where the manipulator must reach red, blue and then yellow targets from ego-centric observations, see Figure 6. We see that ESM again outperforms the LSTM baseline. Full details of the RL setup are given in Appendix A.2.

4.1.4 Structured Memory Applications

The fixed structure of ESM brings additional applications. Two such examples are local obstacle avoidance, and inter-camera transfer learning, see Table 2. We see that policies trained from ego-centric or scene-centric cameras can be freely interchanged without significant loss of performance. This is due to the consistent memory representation adopted. In contrast, the policies of other baselines are not transferable at all. For our obstacle avoidance experiment, we predict the relative target position as an additional auxiliary loss, alongside the selected action, but at test time, we augment the action with a local obstacle avoidant component. Although the most recent depth frame is also a strong signal for local obstacle avoidance, we show that the avoidance based on the full ESM geometry results in fewer collisions when tested on a variant of the drone task with the inclusion of 25 obstacles. This demonstrates one application of utilizing the ESM structure for combining learnt and non-learnt controllers. More details of the obstacle avoidance experiment are given in Appendix A.3.

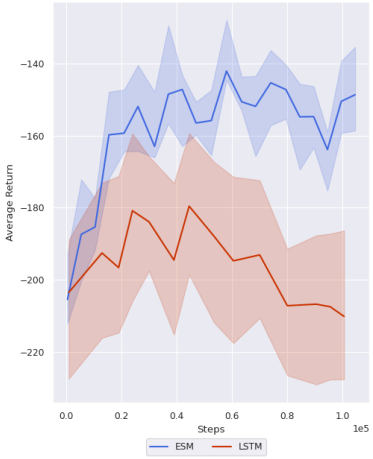

Figure 6: Average return during RL training on sequential reacher task over 5 seeds. Shaded region represent the min and max across trials.

4.2 Object Segmentation

We also explore the suitability of ESMN for object segmentation. One approach is to perform segmentation only in the monocular frame, and then perform fusion based on these probabilities (McCormac et al., 2017). Other works have more recently explored the trade-offs between image-level and map-level modes of inference (Landgraf et al., 2020). We run preliminary experiments to also explore this trade-off, and compare a monocular approach to both ESM and ESMN, where ESM represents map-only inference, while ESMN includes convolutions for both the image-level and map-level inference. We do not attempt to quote state-of-the-art results, but rather to further demonstrate the wide applications of ESMN. We evaluate the object segmentation performance based on the predictions projected to the ego-centric map, and average over the image trajectory. Although the network capacity remains the same between experiments, we see that ESMN outperforms both baselines, effectively learning to combine both modes of inference, see Table 3 for the results, and Figure 7 for a visualization of an ESMN prediction. Further details of this experiment are given in Appendix A.4.

	Mono	ESM	ESMN
Accuracy	50.1%	52.6%	57.0%

Table 3: Object segmentation results on the ScanNet test set for a monocular predictor (Mono), as well as ESM and ESMN.

5 CONCLUSION

Through a diverse set of demonstrations, we have demonstrated that ESMN, and it's off-the-shelf ESM variant, represent a widely applicable computation graph, and trainable module for tasks requiring general spatial reasoning. When compared to other memory baselines for image-to-action learning, our module outperforms these dramatically when learning both from ego-centric images, and when attempting to generalize to different scene camera poses. One weakness of our method is that is assumes the availability of both incremental pose measurements of all scene cameras, and depth measurements, which is not a constraint of the other memory baselines. However, with the ever increasing ubiquity of commercial depth sensors, and with there being plentiful streams for incremental pose measurements including visual odometry, robot kinematics, and inertial sensors, we argue that such measurements are likely to be available in any real-world robotics application. We leave it to future work to further investigate to what extent ESM performance deteriorates with very poor measurements, but with full uncertainty propagation and per pixel fusion, ESM is well suited for accumulating such noisy measurements.

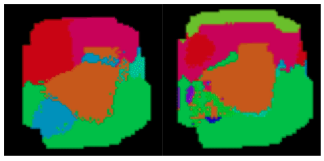

Figure 7: Left: ground truth segmentation labels, and right, predicted labels, both projected into ESM space, for detection on ScanNet images for Eigen13 class labels.

REFERENCES

Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2560–2568, 2018.

Neil Burgess. Spatial memory: how egocentric and allocentric combine. *Trends in cognitive sciences*, 10(12):551–557, 2006.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Cevahir Cigla, Roland Brockers, and Larry Matthies. Gaussian mixture models for temporal depth fusion. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 889–897. IEEE, 2017.

Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J. Davison. Learning to solve nonlinear least squares for monocular stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Péter Fankhauser, Michael Bloesch, Christian Gehring, Marco Hutter, and Roland Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *Mobile Service Robotics*, pp. 433–440. World Scientific, 2014.

Anthony T Fragoso, Cevahir Cigla, Roland Brockers, and Larry H Matthies. Dynamically feasible motion planning for micro air vehicles using an egocylinder. In *Field and Service Robotics*, pp. 433–447. Springer, 2018.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2616–2625, 2017.

Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8476–8484, 2018.

James R Hinman, G William Chapman, and Michael E Hasselmo. Neuronal representation of environmental boundaries in egocentric coordinates. *Nature communications*, 10(1):1–8, 2019.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Stephen James, Marc Freese, and Andrew J Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

Roberta L Klatzky. Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In *Spatial cognition*, pp. 1–17. Springer, 1998.

Zoe Landgraf, Fabian Falck, Michael Bloesch, Stefan Leutenegger, and Andrew Davison. Comparing view-based and map-based semantic labelling in real-time slam. *arXiv preprint arXiv:2002.10342*, 2020.

Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10986–10995, 2019.

John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pp. 4628–4635. IEEE, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136. IEEE, 2011a.

Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pp. 2320–2327. IEEE, 2011b.

Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9226–9235, 2019.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

Emilio Parisotto, H Francis Song, Jack W Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. *arXiv preprint arXiv:1910.06764*, 2019.

Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *International Conference on Intelligent Robots and Systems*. IEEE, 2013.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Cheng Wang, Xiaojing Chen, Heekyung Lee, Sachin S Deshmukh, D Yoganarasimha, Francesco Savelli, and James J Knierim. Egocentric coding of external items in the lateral entorhinal cortex. *Science*, 362(6417):945–949, 2018.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pp. 2397–2406, 2016.

Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 822–838, 2018.

A APPENDIX

A.1 IMITATION LEARNING TRAINING DETAILS

For training, we generate an offline dataset of 100k 16-step sequences from random motions for both environments, and train the agents using imitation learning from known expert actions. For NTM we use a similar variant to that used in ref Wayne et al. (2018), namely, we use sequential appending to the memory, and retroactive updates. We do not cache the memory for later training batches, and so we use a large unroll size of 16 to encourage the baselines to learn over long horizons. We find 16 steps is on average enough time to reach all targets once. We also compare against a network trained on partial oracle omni-directional images, masked at unobserved pixels, which we refer to as Partial-Oracle (PO). We use low-resolution 32×32 images, and a spatial memory size of 180×90. All baselines use the same encoder architecture with features $[16, 32, 64, 128, 192256]$ up to a 1×1 bottleneck, with strides modified depending on whether conditioned on mono or memory images, and are trained for 250k steps with batch size 16 and unroll length 16. We provide the depth information as an image of relative cartesian co-ordinates, bypassing camera intrinsics. We also provide the target descriptor and the incremental pose as tiled images both at conv1 and conv5, and also as a vector for the memory baselines LSTM and NTM.

A.2 REINFORCEMENT LEARNING TRAINING DETAILS

We train both ESM and an LSTM baseline on a similar sequential target reacher task (as defined in Section 4.1) via DQN (Mnih et al., 2015), where the manipulator must reach red, blue and then yellow targets from ego-centric observations. The ESM DQN agent receives omni (180×90) RGB and pointcloud data, while the LSTM baseline receives (128×128) RGB and pointcloud wrist-cam data, with a sequence length of 8. We use discrete delta end-effector translations, with ± 0.05 meters for each axis, with no rotation (resulting in an action size of 6). We use a shaped reward of $r = -len(remaining_targets) - \|e - g\|_2$, where e and g are the gripper and current target translation respectively.

Both methods use a Siamese network to process the RGB and point cloud inputs separately, and consist of 2 convolution (conv) layers with 16 and 32 channels (for each branch). We fuse these branches with a 32 channel conv and 1x1 kernel. The remainder of the architecture then follows similarly to the imitation learning experiments, but we instead use channel sizes of 64 throughout. The network output are 6 values corresponding to the Q-values for each action. Both methods use a learning rate of 0.001, target Q-learning $\tau = 0.001$, batch size of 128, and *leakyReLU* activations. We use an epsilon greedy strategy for exploration that is decayed over 100k training steps from 1 to 0.1.

A.3 Obstacle Avoidance

A visualization of the obstacle avoidance environment is shown in Figure 8. For this experiment, we use a simple local planner which adds a motion vector inversely proportional to the closest depth measurement available. In order to prevent avoidant behaviour away from the targets themselves, we use the auxiliary target location predicted by the network, and prevent depth values within a fixed sphere of this prediction from influencing the obstacle avoidance.

A.4 Object Segmentation

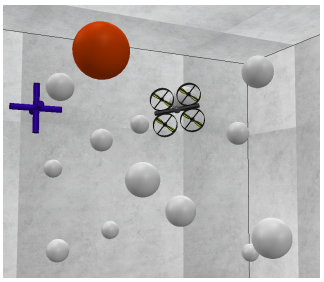

Figure 8: Drone environment with obstacle avoidance

For the object segmentation experiment, we use down-sampled 60×80 images from the ScanNet dataset which we first RGB-Depth align, and an ESM size of 180×90. We use a reduced dataset with frame skip of 30 to minimize the occurrence of frequent near-identical frames. All losses are computed in the ESM frame of reference, either following convolutions for ESM and ESMN, or using the standard ESM RGB projection for the Mono case. All networks use a $[16, 32, 64, 128, 64, 32, 16]$ U-Net architecture, but with each layer doubled with stride 1 for the ESM and Mono cases, making the U-Net $[16, 16, 32, 32, 64, 64, 128, 64, 64, 32, 32, 16, 16]$. Mono applies this before the memory, and ESM applies this after the memory. For ESMN, the U-Net is evenly divided between the monocular image before the memory, and also from the memory to the final prediction, with no stride 1 layers. We also compute ground-truth labels by projecting these through ESM, rather than computing the ground truth Ego-sphere from the complete mesh. Such an approach would also enable semantic predictions in the quantization holes and unobserved regions, which our current approach does not allow, but our current approach was adopted for implementational simplicity.