EMPNet: Neural Localisation and Mapping Using Embedded Memory Points

Gil Avraham   Yan Zuo   Thanuja Dharmasiri

Tom Drummond

ARC Centre of Excellence for Robotic Vision, Monash University, Australia

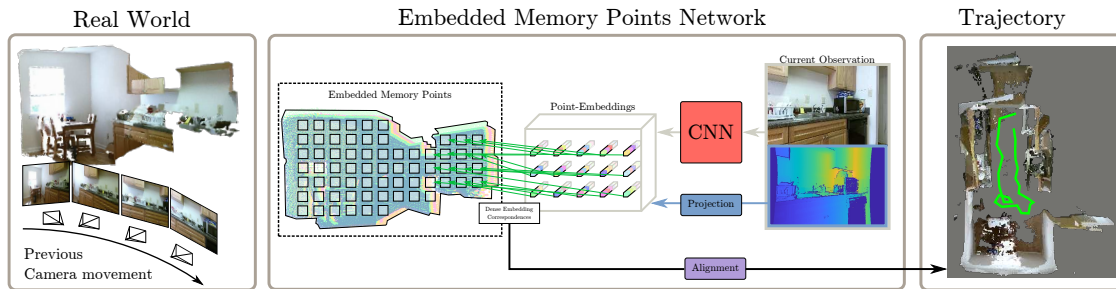{gil.avraham, yan.zuo, thanuja.dharmasiri, tom.drummond}@monash.edu

Figure 1: EMP-Net maintains an internal representation which corresponds to a real world environment. This internal spatial memory is continuously updated through a dense matching algorithm, allowing an autonomous agent to localise and model the world through sequences of observations.

Abstract

Continuously estimating an agent's state space and a representation of its surroundings has proven vital towards full autonomy. A shared common ground among systems which successfully achieve this feat is the integration of previously encountered observations into the current state being estimated. This necessitates the use of a memory module for incorporating previously visited states whilst simultaneously offering an internal representation of the observed environment. In this work we develop a memory module which contains rigidly aligned point-embeddings that represent a coherent scene structure acquired from an RGB-D sequence of observations. The point-embeddings are extracted using modern convolutional neural network architectures, and alignment is performed by computing a dense correspondence matrix between a new observation and the current embeddings residing in the memory module. The whole framework is end-to-end trainable, resulting in a recurrent joint optimisation of the point-embeddings contained in the memory. This process amplifies the shared information across states, providing increased robustness and accuracy. We show significant improvement of our method across a set of experiments performed on the synthetic VIZ-Doom environment and a real world Active Vision Dataset.

1. Introduction

In recent times, there has been a large surge in interest towards developing agents which are fully autonomous. A core aspect of full autonomy lies in the spatial awareness of an agent about its surrounding environment [9]; this understanding would enable the extension towards other useful applications including navigation [10] as well as human-robot interaction [7]. Although performance in image understanding challenges such as segmentation [4, 16, 26], depth estimation [12, 37, 40], video prediction [27, 43], object classification [18, 25, 36] and detection [13, 34] has seen vast improvement with the aid of deep learning, this level of success has yet to translate towards the intersection between spatial awareness and scene understanding. Currently, this is an active area of research [14, 19, 33], with the vision community realising its potential towards merging intelligent agents seamlessly and safely into real world environments.

Fundamentally, an autonomous agent is required to maintain an internal representation of the observed scene structure that may be accessed for performing tasks such

as navigation, planning, object interaction and manipulation [14]. Traditional SLAM methods [9, 29] maintain an internal representation via keyframes, stored in a graph-like structure which provides an efficient approach for large-scale navigation tasks. Although, in order to distil the local structural information of a scene, dense representations [31] are often better suited for the given task. Incidentally, this dense representation is also more applicable for modern deep learning approaches.

Given this, we identify an important relationship between the representation of a scene structure and the geometric formulation of the problem. Nowadays, the increased popularity of cameras with depth sensors mounted on robotic platforms means that RGB-D information of the scene is readily available. For an agent navigating an environment whilst simultaneously collecting colour and depth information, a natural representation is a 3D point entity which can capture the spatial neighbourhood information of its surroundings. The alignment of such representations has been heavily explored in the literature [38].

In this work, we reformulate the task of finding 3D point correspondences as a cross-entropy optimisation problem. By having access to depth sensory and the agent's pose information in the data collection stage, a ground-truth correspondence matrix can be constructed between two consecutive frames such that 3D points which match are assigned a probability of '1', and non-matches assigned a '0'. Using a Convolutional Neural Network (CNN), we extract feature embeddings from an acquired observation, which are then assigned to projected depth points. Collectively, we refer to these embedding-coordinate pairs as point-embeddings. This allows for end-to-end optimisation on the correspondences between closest point-embeddings (Fig. 1).

By iteratively repeating this process, extracted point embeddings stored from previously seen observations are jointly optimised within a processed sequence of frames, forming a recurrent memory mechanism. The point-embeddings along with their 3D location are stored within a memory component which we refer to as the Short-term Spatial Memory Module (SSMM). Through continuously inferring a correspondence matrix between point-embeddings in the SSMM and newly extracted point-embeddings, we obtain the relative pose between the incoming frame and a local coordinate frame of the SSMM. The result is a SSMM which contains point-embeddings which are structurally aligned to their original structure in the real world.

We evaluate our method on two datasets: a synthetic environment from the Doom video-game, and a real environment captured from a robotic platform from the Active Vision Dataset. In both datasets, we show that our method significantly outperforms baselines on localisation tasks.

The rest of this paper is organised as follows: in Section 2, we give a brief review of related work. In Section 3, we provide details of our proposed method. In Section 4, we show experimental results and discuss possible extensions to our method in Section 5.

2. Related Work

The related literature to our work can be organised into three categories.

Frame-based Prior to the introduction of memory based models for localisation and mapping tasks, frame-by-frame methods [8, 23] and more recently [11, 28], explored the exploitation of geometric constraints for reducing the search space when optimising Convolutional Neural Networks (CNN). The pioneering work of [23] applied direct pose regression for inferring the relative pose between two views. The work by [8] enhanced the information provided to the regression network by including the optical flow between two consecutive frames. A natural extension was explored by [11] which simultaneously estimated a depth map along with a latent optical flow constraint for regressing the pose between consecutive frames. CodeSLAM [3] optimises an encoder-decoder setup to efficiently represent depth frames as latent codes. These latent codes are optimised so that pose information can be used to transform one latent code to another. Most recently, [28] combined a photometric loss with depth estimation and additionally used the inferred depth for minimising the residual error of 3D iterative closest point [15] loss. In our work, we similarly minimise a closest point loss, though we minimise the direct closest point errors between an internally modelled environment and an incoming observation.

Sequence-based The importance of maintaining an internal representation of previously seen observations was initially explored in DeepVO [39] and VINet [6]. Both works process sequential information by extracting features using a CNN, which are inputted to an LSTM [20] for fusing past observations while regressing the relative pose between two consecutive frames. DeepTAM [42] reformulated the Dense Tracking and Mapping (DTAM) approach of [31] as a learning problem. Similar to DeepVO, DeepTAM regresses the pose directly and in addition, estimates an expensive cost volume for mapping the environment. An elegant extension to the above approaches by [5] exploits bidirectional LSTMs for obtaining temporally smooth pose estimates (however this bidirectional property introduces an inference lag). Similarly, we maintain a consistent spatial temporal representation of the environment, although our short-term memory recall is more verbose and engineered to have more explicit meaning of localising against previously seen observations.
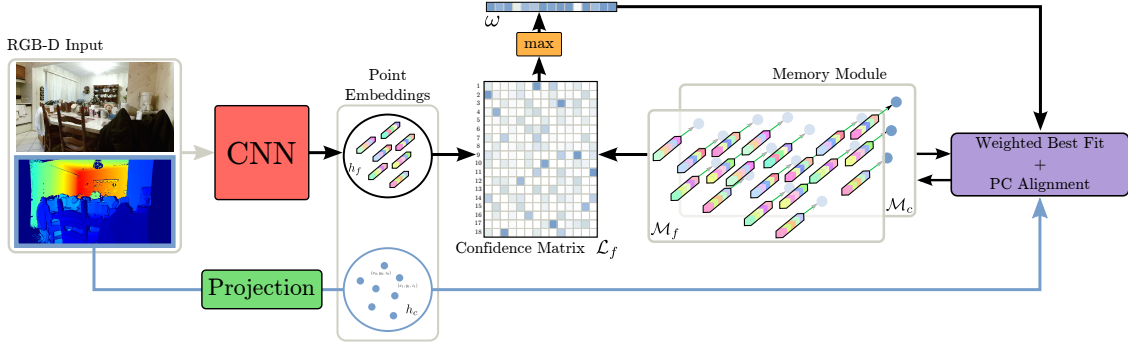
Figure 2: The proposed architecture of EMP-Net. Incoming observations are processed to extract point-embeddings and localise against a short-term memory module. The memory module is updated with the new set of point-embeddings after aligning them to the memory's coordinate frame using the inferred pose.

Map-based Incorporating a more explicit map representation of an agents previously visited states was explored by Reinforcement Learning based approaches [14, 33, 41] where optimising towards a goal which forces an agent to model the environment was found beneficial. Both Neural SLAM [41] and Neural Map [33] have a fixed latent map size, with a 2D top-down map representation. However, both of these works only assess their models on synthetic mazes and toy tasks. [14] extended upon this with the introduction the Cognitive Mapper and Planner (CMP). CMP integrated navigation into the pipeline and also changed the global map representation to an egocentric latent map representation. [19] focused on extending the mapping aspect of [14] through the introduction of MapNet, which learns a ground-projection allocentric latent map of the explored environment. MapNet performs a brute-force localisation process at every time step; by doing so, temporal information is lost and irrelevant areas in the map are considered as viable localisation options. In contrast, our work uses this temporal information as a prior for localisation and updating the internal map.

3. Embedded Memory Points Network

In Fig. 2, a illustrative overview of our system is shown and a brief descriptive summary of our method is provided in the next subsection. Following this, we describe in more detail each core step of our framework. For the remainder of the paper, we use non-bold subscripts to represent matrices or scalars (depending on the context, *i.e.* R), bold subscripts to represent vectors (*i.e.* $\boldsymbol{q}$) and indexing into both is done using brackets (*i.e.* $A[i, j]$ or $\boldsymbol{q}[i]$). Additionally, we refer to the central memory unit of our system, the Short-term Spatial Memory Module (SSMM), as two components, denoted as $\mathcal{M}_f$ and $\mathcal{M}_c$ which indicates the respective stored embeddings and their corresponding 3D points in the SSMM.

3.1. System Overview

An incoming RGB-D observation at time t, $x_t \in \mathbb{R}^{h \times w \times 4}$ of height h and width w, is processed by a CNN (Section 3.2) to produce the embeddings $h_{t,f} \in \mathbb{R}^{N_r \times n}$. Each embedding's corresponding locations in egocentric camera coordinates $h_{t,c} \in \mathbb{R}^{N_r \times 3}$ is obtained through projecting the depth information using the camera intrinsic matrix K. $h_{t,f}$ and $h_{t,c}$ represent the collectively generated point-embeddings. N_r is the number point-embeddings generated and n indicates the number of embedding channels.

Computing pairwise distances between embeddings $h_{t,f}$ and $\mathcal{M}_{t-1,f} \in \mathbb{R}^{N_r b \times n}$, produces the distance map $\mathcal{D}_{t,f} \in \mathbb{R}^{N_r b \times N_r}$ (Section 3.3); with b denoting the buffer size of $\mathcal{M}$. The distance map $\mathcal{D}_{t,f}$ is converted into a Confidence Map $\mathcal{L}_{t,f} \in \mathbb{R}^{N_r b \times N_r}$ by applying a column-wise softmax operation and obtains the weight vector $\boldsymbol{\omega}_t \in \mathbb{R}^{N_r}$. This allows the system to optimise for the relative pose $T_t \in \mathbb{SE}(3)$, between the downsampled point cloud $h_{t,c}$ and their corresponding matches in $\mathcal{M}_{t,c} \in \mathbb{R}^{N_r b \times 3}$ in a weighted least squares formulation (Section 3.4).

Finally, an update step is performed by populating $\mathcal{M}_{t-1,f}$ with $h_{t,f}$ and transforming the downsampled point cloud $h_{t,c}$ in egocentric coordinate frame to $\mathcal{M}_{t-1,c}$'s coordinate frame by applying the estimated pose T_t on $h_{t,c}$ and populating $\mathcal{M}_{t-1,c} \in \mathbb{R}^{N_r b \times 3}$, resulting in an updated $\mathcal{M}_{t,f}$ and $\mathcal{M}_{t,c}$. For the rest of the paper, for reducing clutter the time subscript t will be omitted unless specified otherwise.

3.2. Extracting Point Embeddings

To extract point-embeddings from observations, we use a CNN architecture which receives an RGB-D input $x \in \mathbb{R}^{h \times w \times 4}$ and produces a tensor $x' \in \mathbb{R}^{h' \times w' \times n}$ where $h' < h, w' < w$, and n is the channel length of each embedding. At this stage, we need to associate an embedding

in $\mathbf{x}'[i,j,.]$ with the 3D point it represents. This is accomplished in the following manner: first, the given depth map $D \in \mathbb{R}^{h \times w}$ is resized to $D' \in \mathbb{R}^{h' \times w'}$ such that it matches the spatial dimensions of x'. In this case, a traditional bilinear downsampling approach was found to be sufficient. Next, we compute the 3D location of each entry in $D'[i,j]$ in egocentric camera coordinates using the known camera intrinsic matrix K and the downsampled depth map D' as shown below:

$$P_c[i,j,k] = D'[i,j]K^{-1}\left[i,j,1\right]^{\top} \qquad (1)$$

where $P_c \in \mathbb{R}^{h' \times w' \times 3}$. An entry in $P_c[i,j,k]$ is a 3D point in an egocentric camera coordinate frame corresponding to the depth map entry $D'[i,j]$. Rearranging $x' \in \mathbb{R}^{h' \times w' \times n}$ to $h_f \in \mathbb{R}^{N_r \times n}$ with $N_r = h'w'$ and similarly P_c to $h_c \in \mathbb{R}^{N_r \times 3}$ yields two initial inputs to the next component, which localises both h_f, h_c against $\mathcal{M}_f, \mathcal{M}_c$ that contains previously stored embeddings and 3D points in $\mathcal{M}_c$'s coordinate frame.

3.3. Short-term Spatial Memory Localisation

We require the output of the CNN, h_f, to endow the framework with embeddings that can coherently match another set of point embeddings contained in $\mathcal{M}_f$. For this, we develop a loss function stemming from the ICP algorithm [2]. Finding the relative pose between two sets of point clouds requires finding the matching correspondences between them. Typically, two key difficulties emerge from this task: points not having any correspondences (due to partial overlaps), and points having correspondences which have more certainty than others. Here, we formulate the optimisation problem to address both issues using a unified weighting approach.

For an incoming observation x, we extract point-embeddings h_f in the manner described in Section 3.2. We define the following operation as taking pairwise distances between the embeddings $\mathcal{M}_f$ and h_f:

$$\mathcal{D}_f[i,j] = d_{\phi}(\mathcal{M}_f[i,.], h_f[j,.]) \qquad (2)$$

Where $\mathcal{M}_f[i,.], h_f[j,.] \in \mathbb{R}^n$ are embedding row vectors, $\mathcal{D}_f \in \mathbb{R}^{N_r b \times N_r}$ is the pairwise distances matrix for the embeddings and d_{ϕ} is a distance metric on the embedding space. Reformulating $\mathcal{D}_f$ by applying the softmax operation yields:

$$\mathcal{L}_f[i,j] = \frac{e^{-\mathcal{D}_f[i,j]}}{\sum_{i'=1}^{N_r b} e^{-\mathcal{D}_f[i',j]}} \qquad (3)$$

Where $\mathcal{L}_f \in \mathbb{R}^{N_r b \times N_r}$ is the confidence matrix between embeddings in $\mathcal{M}_f$ and h_f. Note that a single column vector $\mathcal{L}_f[.,j]$ is a confidence vector between the point $h_c[j,.]$ and the entire set of points in $\mathcal{M}_c$ (Fig. 3 illustrates this operation).
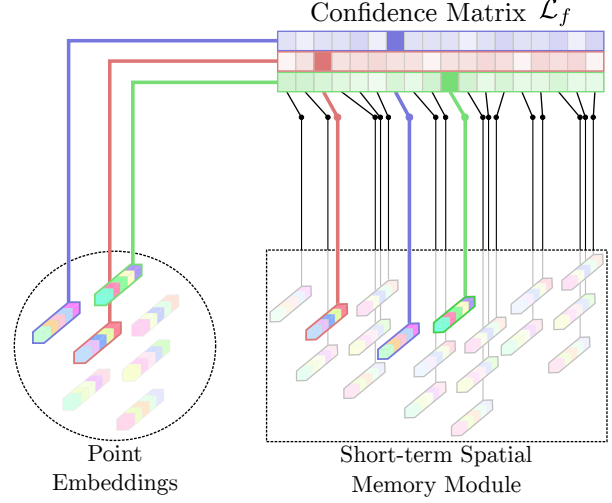

Figure 3: The confidence matrix is constructed by densely computing a distance between embedded points of a current observation and the stored embedded memory points in the Short-term Spatial Memory Module. A darker colour indicates a higher confidence value in the confidence matrix.

For optimising the confidence matrix $\mathcal{L}_f$ towards a ground-truth confidence matrix $\mathcal{L}_{gt}$, we define our loss function as the cross-entropy loss:

$$loss_c = -\frac{1}{N_r} \sum_{j=1}^{N_r} \sum_{i=1}^{N_r b} \mathcal{L}_{gt}[i,j] \log \mathcal{L}_f[i,j] \qquad (4)$$

The ground-truth confidence matrix $\mathcal{L}_{gt} \in \mathbb{R}^{N_r b \times N_r}$ is computed using a procedure similar to the one outlined above; more explicitly, we define $\mathcal{M}_{gt} \in \mathbb{R}^{N_r b \times 3}$ to be a sequence of point clouds, aligned using the ground-truth poses to a shared coordinate frame. For an incoming ground-truth aligned point cloud $h_{gt} = T_{gt}h_c$ at time t which follows point sequences stored in $\mathcal{M}_{gt}$, the ground-truth pairwise distances are computed as follows:

$$\mathcal{D}_{gt}[i,j] = \|\mathcal{M}_{gt}[i,.] - h_{gt}[j,.]\|_2 \qquad (5)$$

Where $\mathcal{M}_{gt}[i,.], h_{gt}[j,.] \in \mathbb{R}^3$ are row vectors. A property of $\mathcal{D}_{gt}$ is that points which are close enough will have a small distance value whilst points which do not have a match (*i.e.* are in a non-overlapping region) will have a large distance. This can be exploited in a way which will amplify both matching and non-matching cases, where a probability '1' is assigned to matches and a '0' to non-matches. Similar to Eq. 3, we reformulate matrix $\mathcal{D}_{gt}$:

$$\mathcal{L}_{gt}[i,j] = \frac{e^{-\tau \mathcal{D}_{gt}[i,j]}}{\sum_{i'=1}^{N_r b} e^{-\tau \mathcal{D}_{gt}[i',j]}} \qquad (6)$$

The temperature coefficient τ controls the amplification of distance correspondences and is a hyper-parameter. Finally,

we note the operations discussed in this section are naturally parallelised and can be computed efficiently using modern GPU architectures.

3.4. Best-fitting of Weighted Correspondences

In the previous sections, we formulated a loss which optimises the embeddings h_f to follow the 3D closest point criteria. This allows for the recovery of a matrix $\mathcal{L}_f$, which contains confidence values between a point in h_c and $\mathcal{M}_c$. These confidence values represent weights that can be used for applying a weighted best-fit algorithm. The weights are obtained as follows:

$$\omega[j] = \max_{i'} \mathcal{L}_f[i', j] \tag{7}$$

and respectively, the point index in $\mathcal{M}_c$ corresponding to point $h_c[j, .]$:

$$c[j] = \arg\max_{i'} \mathcal{L}_f[i', j] \tag{8}$$

Where $c \in \mathbb{R}^{N_r}$ is the indexing vector for aligning the correspondences in $\mathcal{M}_c$ to those in h_c. The weights in vector $\omega \in \mathbb{R}^{N_r}$ are the respective confidences for those matches. Computing the relative-pose between the point cloud h_c in egocentric coordinates and the points in $\mathcal{M}_c$, with its respective coordinate frame, can be estimated using a weighted best-fit approach.

Weighted Best-fitting Given two sets of point clouds $p \in \mathbb{R}^{M \times 3}$, their correspondences $q \in \mathbb{R}^{M \times 3}$ and weight vector $\omega \in \mathbb{R}^M$. The rigid-transform can be computed in a closed form and is optimal in a weighted least-squares sense (proof in [15]). Formally, we solve:

$$R, \mathbf{t} = \arg\min_{R \in SO(3), \mathbf{t} \in \mathbb{R}^3} \sum_{\ell=1}^{M} \omega[\ell] \left\| \mathbf{q}[\ell, .] - (R\mathbf{p}[\ell, .] - \mathbf{t}) \right\|_2 \tag{9}$$

For obtaining $R, \mathbf{t}$ we initially compute:

$$\bar{\mathbf{p}} = \frac{\sum_{\ell=1}^{M} \omega[\ell]\mathbf{p}[\ell, .]}{\sum_{\ell=1}^{M} \omega[\ell]}, \quad \bar{\mathbf{q}} = \frac{\sum_{\ell=1}^{M} \omega[\ell]\mathbf{q}[\ell, .]}{\sum_{\ell=1}^{M} \omega[\ell]} \tag{10}$$

With $\bar{\mathbf{p}}, \bar{\mathbf{q}} \in \mathbb{R}^3$ being the weighted average centroids of p, q respectively. By subtracting each weighted centroid from its respective point cloud we get:

$$\hat{\mathbf{p}}[\ell, .] = \mathbf{p}[\ell, .] - \bar{\mathbf{p}}, \quad \hat{\mathbf{q}}[\ell, .] = \mathbf{q}[\ell, .] - \bar{\mathbf{q}} \tag{11}$$

Finally, by defining $\Omega = diag(\omega)$ and applying SVD decomposition such that: $U\Sigma V^\top = \hat{p}^\top \Omega \hat{q}$, the rotation matrix R is computed as:

$$R = V diag(1, 1, \det(VU^\top))U^\top \tag{12}$$

and the translation vector $\mathbf{t}$:

$$\mathbf{t} = \bar{\mathbf{q}} - R\bar{\mathbf{p}} \tag{13}$$

By performing the weighted best-fit procedure outlined above, we obtain the relative pose T between h_c in egocentric coordinate frame and $\mathcal{M}_c$ in its respective coordinate frame. In the last step of our framework, h_c is transformed using the estimated pose yielding $h_c^{'} = Th_c^\top$, where populating $\mathcal{M}_c$ with $h_c^{'}$ and $\mathcal{M}_f$ with h_f need not be in a particular order, as both are unstructured.

A straight-forward extension to this approach is imposing pose regularisation on the loss developed in Section 3.3. This is achieved through projecting $\mathcal{M}_c$ onto the confidence matrix $\mathcal{L}_f$:

$$\bar{\mathcal{M}}_c = \mathcal{L}_f^\top \mathcal{M}_c \tag{14}$$

with $\bar{\mathcal{M}}_c \in \mathbb{R}^{N_r \times 3}$ being the correspondences of h_c. The best-fit approach (without weights ω) from Eq. 9 is applied to obtain the rotation matrix R and translation vector $\mathbf{t}$. This alternative formulation also makes our method fully differentiable. Finally, we modify the loss given in Eq. 4 by adding two regularisation terms:

$$Loss = loss_c + \lambda_R loss_R + \lambda_t loss_t \tag{15}$$

both $loss_R$ and $loss_t$ are formulated as in [23]. In summary, both approaches formulated differ by how the confidence matrix $\mathcal{L}_f$ is used to compute the pose (Eq. 4 and Eq. 15).

4. Experiments

To compare our models qualitatively and quantitatively, we perform experiments on two challenging benchmarks: a synthetic environment (VIZDoom [22]) and a real world indoor environment (Active Vision Dataset [1]). We evaluate two variants of our model: EMP-Net which optimises Eq. 4 and EMP-Net-Pose which optimises Eq. 15. For the Doom dataset, we compared our framework against DeepVO [39] and MapNet [19], both which maintain an internal representation of previously seen observations. Additionally, we compared our models against a recent state-of-the-art frame-to-frame approach ENG [11]. For the AVD dataset, we also compare with a mature classic SLAM baseline, an RGB-D implementation of ORB-SLAM2 [29].

Network Architecture For extracting feature embeddings, we employ a U-Net architecture [35]. We initialised our network weights using the initialisation scheme detailed in [17]. The U-Net uses an encoder-decoder setup where the encoder consists of three encoder blocks separated by max pooling layers and the decoder consists of two decoder blocks. Each block in the encoder consists of two sequences

of convolution layers comprising of 3×3 filters, followed by Batch Normalisation [21] and ReLU Activation [30]. Each block in the decoder consists of a transposed convolution layer followed by Batch Normalisation [21], ReLU Activation [30] followed by a convolution layer. The transposed convolution layer upsamples the input using a stride 2 deconvolution, the output of which is concatenated to its matching output from the encoder block.

Training Settings For both synthetic and real experiments, our EMP-Net model is trained with a batch size of 16, where every instance within a batch is a sequence of 5 consecutive frames resized to 120×160. The RGB and depth data were scaled to between $[0, 1]$. The buffer size of the SSMM is $b = 4$ and the number of extracted point-embeddings is $N_r = 4800$. The temperature parameter was tested with values between $\tau = [10^3, 10^6]$, where we found the model to be fairly invariant to this value. In all of the experiments shown $\tau = 10^5$. The embedding distance function is defined as the $L2$ distance, $d_\phi = \|a - b\|_2$. $\lambda_t = 0.02$ and $\lambda_R = 5$ are chosen to maintain the same ratio as in [23]. We use the ADAM optimiser [24], using the default first and second moment terms of 0.9 and 0.999 values respectively. We use a learning rate of 10^{-3} and train for 10 epochs.

Error Metrics Across both datasets, we quantitatively benchmark against baselines using two error metrics. We measure the Average Position Error (APE), which denotes the average Euclidean distance between predicted position of the agent and a corresponding ground-truth position. Additionally, we inspect the Average Trajectory Error (ATE) which describes the minimum RMS error in position between a translated and rotated predicted trajectory *w.r.t* a ground-truth trajectory. Thus, for longer sequences the APE will naturally be worse as it does not correct for drifts occurring over time. Similar to [19], we measure both short-term APE over 5 observation frames (APE-5) as well as long term APE (APE-50) and ATE (ATE-50) over 50 observation frames.

4.1. Synthetic 3D Data

We used VIZDoom to record human players performing 4 speed-runs of the game with in-game sprites and enemies turned off. Despite VIZDoom being a synthetic environment it provides rich and complex visual scenarios that emulate the difficulties encountered in real world settings. The captured recordings include RGB-D and camera pose data, which correspond to 120k sequences. Training sequences are composed of 5 frames, sampled every second frame of recorded video. For testing, we randomly select sequences of 50 consecutive frames and remove those from the training set construction.
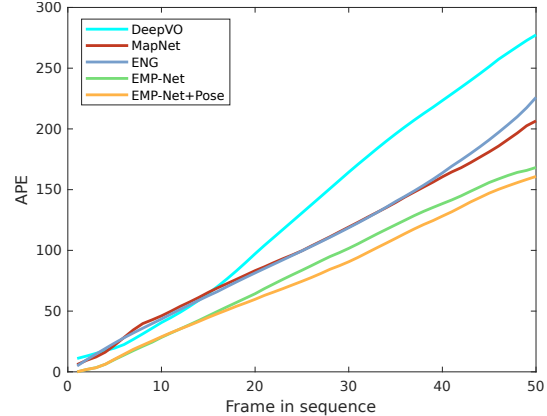

Figure 4: Average Positional Error (APE) over different sequence lengths (5-50 frames) on the Doom dataset

Doom data [22]	APE-5	APE-50	ATE-50
DeepVO [39]	19.56	277.4	111
MapNet [19]	21.98	206.6	76
ENG [11]	23.71	225.9	105
EMP-Net (Ours)	**10.10**	168.3	68
EMP-Net-Pose (Ours)	10.45	**160.9**	**59**

Table 1: Average Position Error (APE) and Absolute Trajectory Error (ATE) on VIZDoom dataset.

Quantitative Results We measure APE across the test sequences on all the above mentioned models (Fig. 4). Increasing the sequence length beyond a sequence length of 5 examines its ability to generalise beyond the length of the training sequences. Both DeepVO [39] and ENG [11] lack an internal map representation to localise against and similarly both methods suffer from a larger accumulated drift towards the end of the sequence. MapNet [19] fairs better, although suffers from inaccuracies due to cell quantisation and false pose modalities which appear over longer periods of time. We note that both our non-regularised and regularised methods (EMP-Net and EMP-Net-Pose) significantly outperform the compared baselines across all sequence lengths. The two variants are similar in their performance, with a marginal improvement that is gained by the additional pose regularisation. This minor improvement is explained from the nature of the data. VIZDoom provides noiseless depth and pose information which correspond perfectly to each other, as both are obtained directly from the game engine. In other words, the provided ground-truth labels for regressing the confidence matrix contain all the necessary information regarding the pose. In Table 1, we show the APE across observation sequences of 5 frames and 50 frames, as well as the ATE across a 50 observation sequence.

Observations corresponding to fixed memory

(a)

Observations at varying camera displacements

Low confidence   High confidence
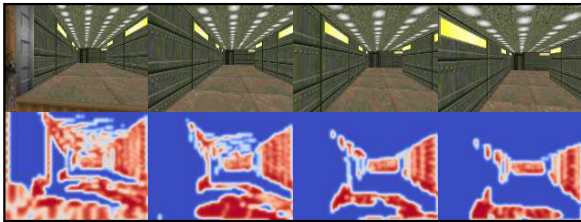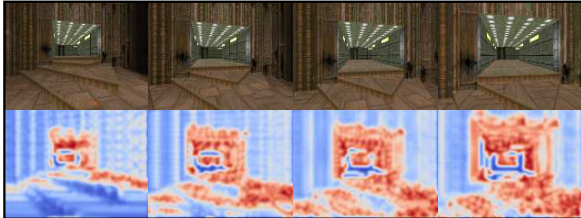
(b)

Figure 5: Qualitative results of Section 4.2. (a): Observations corresponding to the embedded memory points stored in EMP-Net's memory module. (b): Incoming observations which are processed and localised against the stored embedded memory. For distinguishable landmarks (*i.e.* stairs and corridor entrance) confidence values are higher. In later frames, areas that are not visible in the past observations are correctly receiving near zero confidence.

4.2. Confidence Matrix Interpretation

In this section, we provide additional insight about the inferred confidence matrix in EMP-Net. We run an experiment which allows the system to process test sequences and store point-embeddings up to the size of the buffer (*i.e* four observation frames). Beyond this point, we discontinue storing point embeddings but continue to process the sequence by localising against the existing embeddings within the SSMM. This allows for simulation of large camera motions and assesses the robustness of the estimated confidence matrix across increasingly larger camera motions. The top figure in Fig. 6 shows the APE of EMP-Net as the camera baseline grows. For reference, we plot the APE for a standard ICP [15] point-to-point implementation. Note that while the APE value increases as we shift further away from the baseline, EMP-Net is demonstrably more ro-
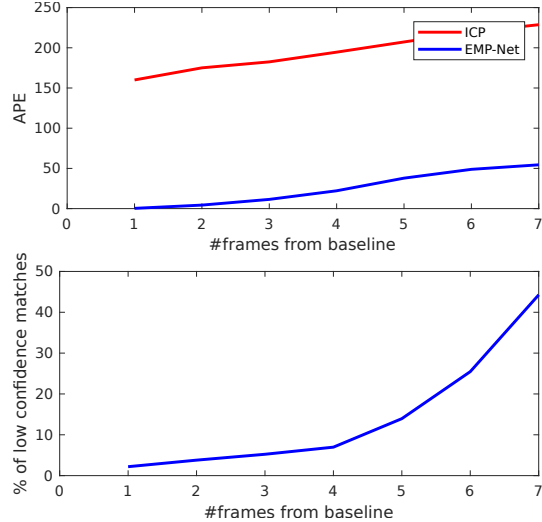

Figure 6: **Top**: APE measured over increasing camera baselines on the Doom dataset. For reference we compare to traditional ICP algorithm [15]. **Bottom**: Percentage of low confidence matches over increasing baselines. In this case, low confidence indicates a confidence of less than 0.05. As the baseline increases, the agent becomes increasingly less confident.

bust to larger shifts in the camera baseline. The reduced performance on APE correlates with lower confidence of the system as evidenced by the bottom figure in Fig. 6.

In Fig. 5b, we show confidence heat maps along with their corresponding observation frames computed against a fixed memory from observations in Fig. 5a. These confidence heat maps are obtained by reshaping the confidence weight vector ω to the size of the downsampled observation frame (60×80). Note that higher confidence is assigned towards landmarks with distinguishable features (*e.g.* stairs, corridor entrance, *etc.*), whilst lower confidence is assigned to low texture landmarks (*e.g.* walls and floor). For frames with little overlap, the system assigns high confidence to landmarks that it is able to locate in its memory buffer (*i.e.* visible in "fixed memory" frames). At times, these confidences may be overestimated due to a lack of better correspondences from its memory.

4.3. Real World Data

For our real world data experiments, we use the Active Vision Dataset (AVD) [1]. This dataset consists of RGB-D images across 19 indoor scenes. Images are captured by a robotic platform which traverses a 2D grid with translation steps size of 30cm and $30°$ in rotation. For generating robot navigation trajectories, the captured images can be arbitrarily combined. Similar to [19], for training, we sampled 200,000 random trajectories, each consisting of 5
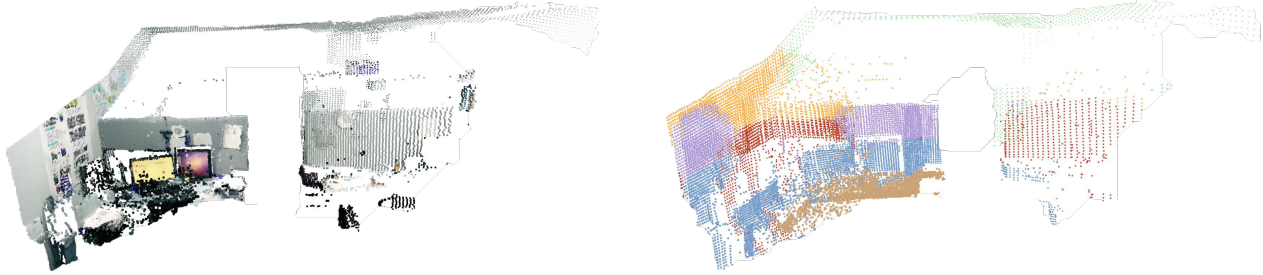
Figure 7: Inspecting the point embedding values provides insight of the underlying operation of EMP-Net. **Left**: The ground-truth alignment of a sequence of 5 frames. **Right**: The corresponding downsampled point cloud in the *SSMM*. The colors define cluster centers of the embeddings. We observe a mixture of spatial and semantic segmentation that is learned without explicit supervision.

frames, where the trajectory was chosen using the shortest path between 2 randomly selected locations from 18 out of the 19 provided scenes. For testing, we sampled 50 random trajectories, each consisting of 50 frames, where the trajectory was chosen from the unseen test scene.

Quantitative and Qualitative Results   We measure APE across the test sequences and show results in Fig. 8. Once again, we increase the sequence length for testing to sequences beyond 5 observation frames to evaluate the ability to generalise beyond training sequence length. Both the non-regularised and regularised methods of EMP-Net significantly outperform the compared baselines across all sequence lengths. In this case, unlike the VIZDoom environment, the use of real world data is accompanied with noisy sensory measurements. Consequently, EMP-Net-Pose is observably more robust than its non-regularised version. In Table 2, we show the APE across observation sequences of 5 frames and 50 frames, as well as the ATE across a 50 observation sequence for the test AVD dataset.

In Fig. 7 we provide additional insight on interpreting the information contained in the learned embeddings of EMP-Net. A snapshot of the SSMM can be seen in Fig. 7 (Right), where we show a downsampled point cloud stored in the SSMM with the inferred aligning. Each point in the SSMM has a corresponding embedding vector. The colour assigned to each point is a cluster centroid colour code that was obtained by performing a *k-means* clustering over the embeddings. A mixture of spatial and semantic segmentation can be observed. For reference, Fig. 7 (Left) is the ground-truth alignment of the point clouds obtained at the original resolution with their corresponding RGB values. For additional qualitative results on the AVD dataset, please refer to our supplementary video material.

5. Future Work

In future work, we look towards extending EMP-Net to larger navigation problems by addressing the linear com-
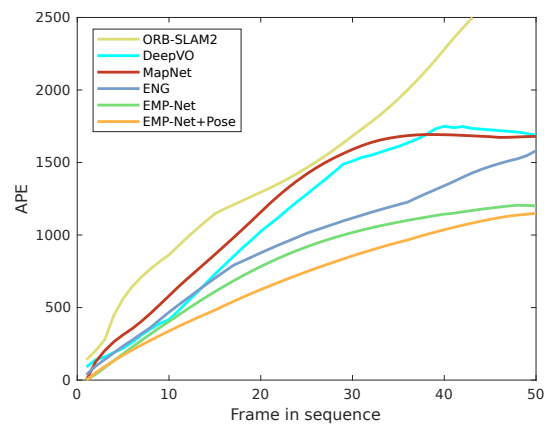

Figure 8: Average Positonal Error (APE) over different sequence lengths (5-50 frames) on the Active Vision Dataset

AVD data [1]	APE-5	APE-50	ATE-50
ORB-SLAM2 (RGB-D) [29]	432	3090	794
DeepVO [39]	220.0	1690	741
MapNet [19]	312.3	1680	601
ENG [11]	234.3	1582	757
EMP-Net (Ours)	181.6	1201	381
EMP-Net-Pose (Ours)	**171.8**	**1150**	**360**

Table 2: Average Position Error (APE) and Absolute Trajectory Error (ATE) on the Active Vision Dataset.

plexity growth of computing the correspondence matrix (*i.e.* large buffer sizes). Extensions worth pursuing for reducing this complexity are non-dense methods for generating correspondences by using approximate nearest neighbour search like methods or formulating the vocabulary tree [32] so it can be integrated within modern deep learning frameworks.

References

[1] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Košecká, and Alexander C Berg. A dataset for developing and benchmarking active vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1378–1385. IEEE, 2017. 5, 7, 8

[2] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987. 4

[3] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslamlearning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2560–2568, 2018. 2

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. 1

[5] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6856–6864, 2017. 2

[6] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 2

[7] Peter I Corke. Visual control of robot manipulators–a review. In *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, pages 1–31. World Scientific, 1993. 1

[8] Gabriele Costante, Michele Mancini, Paolo Valigi, and Thomas A Ciarfuglia. Exploring representation learning with cnns for frame-to-frame ego-motion estimation. *IEEE robotics and automation letters*, 1(1):18–25, 2016. 2

[9] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE, 2003. 1, 2

[10] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007. 1

[11] Thanuja Dharmasiri, Andrew Spek, and Tom Drummond. Eng: End-to-end neural geometry for robust depth and pose estimation using cnns. *arXiv preprint arXiv:1807.05705*, 2018. 2, 5, 6, 8

[12] David Eigen, Christian Puhrsch, and Rob 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. 1

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[14] 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*, pages 2616–2625, 2017. 1, 2, 3

[15] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2, 5, 7

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 5

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[19] 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*, pages 8476–8484, 2018. 1, 3, 5, 6, 7, 8

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 6

[22] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016. 5, 6

[23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 2, 5, 6

[24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[27] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. 1

[28] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018. 2

[29] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 2, 5, 8

[30] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 6

[31] 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*, pages 2320–2327. IEEE, 2011. 2

[32] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006. 8

[33] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017. 1, 3

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[37] Andrew Spek, Thanuja Dharmasiri, and Tom Drummond. Cream: Condensed real-time models for depth prediction using convolutional neural networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 540–547. IEEE, 2018. 1

[38] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217, 2013. 2

[39] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017. 2, 5, 6, 8

[40] Chamara Saroj Weerasekera, Thanuja Dharmasiri, Ravi Garg, Tom Drummond, and Ian Reid. Just-in-time reconstruction: Inpainting sparse maps using single view depth predictors as priors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 1

[41] Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017. 3

[42] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018. 2

[43] Yan Zuo, Gil Avraham, and Tom Drummond. Traversing latent space using decision ferns. In *Asian Conference on Computer Vision*, pages 593–608. Springer, 2018. 1