Learning Composable Behavior Embeddings for Long-horizon Visual Navigation

Xiangyun Meng, Yu Xiang and Dieter Fox

***Abstract*— Learning high-level navigation behaviors has important implications: it enables robots to build compact visual memory for repeating demonstrations and to build sparse topological maps for planning in novel environments. Existing approaches only learn discrete, short-horizon behaviors. These standalone behaviors usually assume a discrete action space with simple robot dynamics, thus they cannot capture the intricacy and complexity of real-world trajectories. To this end, we propose *Composable Behavior Embedding* (CBE), a continuous behavior representation for long-horizon visual navigation. CBE is learned in an end-to-end fashion; it effectively captures path geometry and is robust to unseen obstacles. We show that CBE can be used to performing memory-efficient path following and topological mapping, saving more than an order of magnitude of memory than behavior-less approaches.**

I. Introduction

Humans spend significant amounts of time navigating between familiar locations: grabbing a cup of coffee from the kitchen, going to the printer room to collect papers, or retrieving mail from the mailbox. These navigation routines are executed with little conscious effort because they are so repetitive that they have almost become part of our muscle memory. This saves cognitive load, allowing us to concentrate on more important tasks. From a robot learning perspective, enabling a robot to perform such navigation routines robustly with minimal guidance is beneficial, because it saves memory, speeds up computation, and opens up opportunities to construct sparse, persistent memory of environments for efficient planning and control.

Learning high-level behaviors or skills for robots has become an important area of research recently. Most existing works focus on synthetic control tasks or fixed workspace manipulation tasks [1], [2], [3], [4], where environments are fully observable and a robot can be position-controlled. This is hardly applicable to egocentric visual navigation, where the environment is partially observable, ground truth state may not be available, and the robot may have non-holonomic motion constraints. Because of this, most recent works use predefined behaviors [5], [6], [7], with a few attempts on unsupervised or self-supervised behavior learning [8], [9], [10]. However, these behaviors are usually short-horizon (e.g., "turn left"), discrete, and cannot follow precise specifications (e.g., distance to go or angle to turn). Due to these limitations, they are not able to encode *complex and long-horizon* behaviors in a general fashion, such as when following an instruction "go towards northeast by about 5 meters and then turn right to follow the hallway till the end". This limits their applicability in constructing sparse visual memory for downstream navigation tasks.

Xiangyun Meng and Dieter Fox are with the Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA {xiangyun, fox}@cs.washington.edu

Yu Xiang and Dieter Fox are with NVIDIA, Seattle, WA 98105, USA {yux, dieterf}@nvidia.com

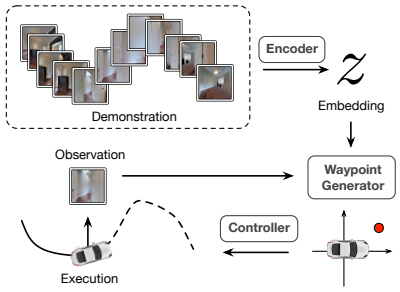


Fig. 1: High-level overview of behavior learning and execution. CBE learns to embed image sequences such that it can replicate a demonstrated trajectory using visual, closed-loop waypoint generation and control.

To address this problem, we propose *Composable Behavior Embedding*, a robot-agnostic behavior representation for visual navigation (Figure 1). At its core is a behavior encoder that compresses a high-dimensional visual demonstration sequence into a low-dimensional embedding. During execution, a waypoint generator is conditioned on the embedding and current observation to generate local waypoints for a low-level controller to replicate the demonstration. The embeddings are learned in an end-to-end fashion by minimizing the waypoint reconstruction loss. It effectively learns to extract path geometry from visual demonstrations, making it generalize extremely well to novel environments.

CBE has two desired properties: i) it is compact. The embedding is only 32-dimensional, allowing a robot to build visual memory an order of magnitude smaller than existing approaches [11], [12], and ii) it is composable. A robot can robustly follow a long path via behavior segmentation, or combine embeddings from multiple demonstrations to perform goal-directed navigation tasks.

SLAM [13], [14] is a competitive alternative for building visual memories. CBE has several advantages over SLAM: i) CBE is more than $10x$ efficient at encoding demonstrations than ORB-SLAM [13]; ii) CBE works with low-resolution images where SLAM breaks down, allowing it to be deployed on miniature robots without high-quality cameras; iii) CBE has a simpler design with few tuning parameters and is end-to-end trainable. Hence, CBE is an attractive approach towards building a robust and efficient learning-based visual navigation system.

We show how the embeddings generated by CBE enable a non-holonomic robot to reach goals more than 150 time

steps away with no intermediate guidance, even when unseen obstacles are present. We further illustrate how the learned embeddings can be applied to two downstream tasks: one-shot trajectory following and topological mapping. We show that with the learned embeddings we can build visual memory an order of magnitude smaller than existing approaches for these downstream tasks. We conduct detailed quantitative and qualitative analysis to verify our design decisions and how it is compared to a variety of baselines.

II. Related Work

Visual Navigation. Classical navigation systems rely on building a metric map from laser scans or visual images for robust state estimation, planning, and control [15], [14]. Recent advances in visual navigation move towards non-metric, learning-based methods, such as short-horizon goal-directed navigation [16], path following [12], [17], [6], or building a cognitive mapping system for planning [18], [19], [11], [20]. Solving long-horizon navigation tasks requires some form of visual memory [18], [11], [21], [12], [17]. Due to visual occlusion, dense observations have to be stored, making it difficult to scale to large environments. Our main contribution is to learn a compact embedding so that a robot only stores a sparse set of visual features. These embeddings serve as sparse visual memory for diverse downstream tasks, such as path following and topological mapping.

Learning from Demonstrations. Perhaps the most direct approach to learn from demonstrations is imitation learning [22], [23]. Imitation learning learns fixed policies that are hard to generalize to novel tasks. Recent works learn latent distributions to encode a diverse skill set [24], [25], [1], [3]. These works focus on manipulation tasks in a fully observable workspace, and hence they cannot generalize to novel, partially observable environments as in indoor navigation. Contrary to existing works that hardcode environments into the skills, we learn a common behavior manifold, allowing a robot to adapt to new environments quickly.

Unsupervised Skill Learning. Learning high-level skills help to solve long-horizon tasks more effectively. However, most works on skill learning assume fully observable state spaces in known environments [26], [27], [28], [4], [29]. This is not applicable in egocentric visual navigation, where environments are partially observable and no ground truth robot state is available. So far only discrete, short-horizon navigation skills can be learned [30], [9], and these skills have only been used for exploration and point-goal navigation tasks. To the best of our knowledge, we are the first to show that diverse and long-horizon navigation skills can be effectively learned from visual data. More importantly, we show that these skills can serve as building blocks for constructing a sparse persistent spatial memory for navigating in novel environments.

Sequence-to-Sequence Models. Our method is inspired by seq2seq models, which have been widely used in language processing [31], [32] and trajectory prediction [33], [34]. An important distinction is that we save the latent states as part of the visual memory. Moreover, our decoding process generates controls that are conditioned on the current rollout, which is essential for correcting drift and avoiding obstacles.

III. Composable Behavior Embedding (CBE)

A. Overview

We consider a goal-directed navigation task where a robot needs to navigate from its current location s to a goal g. We assume that a demonstration containing a sequence of RGB observations $o_1, o_2, ..., o_T$ connecting s and g is given to the robot. Since the trajectory can be long and complex, intermediate information needs to be memorized to help the robot follow the demonstration [12], [17], [11].

Similar to [35], our navigation system guides the robot by generating *relative waypoints* that are used by a low-level controller to compute motor commands (e.g., velocity and steering angle). To follow a demonstrated trajectory, the robot could use visual control to match its observations against the sequence of demonstrated observations as in [11]. However, such an approach is highly memory inefficient, since it requires rather densely stored images. To overcome this problem, CBE encodes the sequence of visual observations $o_1, o_2, ..., o_T$ into a low-dimensional behavior embedding z_D (Figure 1). During execution, at each time step, a waypoint generator uses z_D and the current observation to produce a waypoint for the low-level controller. Both state and action space are continuous, and the system operates in a closed-loop fashion to correct any drift due to noise and non-holonomic motion constraints. Different demonstrations and even their executions can be of different lengths.

For very long and complex trajectories, a single z is insufficient due to compounding errors. To address this problem, we segment long trajectories into sequences of embeddings, interleaved by visual attractors for state calibration. The segmented behaviors are used for solving downstream navigation tasks, which we detail in Sec V.

B. Learning Continuous Navigation Behaviors

The behavior encoder $\mathbb{B}_{\text{enc}}$ (left half of Fig. 2) maps observation streams $o_1, o_2, o_3, ..., o_T$ into a low-dimensional embedding $z_D = \mathbb{B}_{\text{enc}}(o_1, o_2, ..., o_T)$. To do so, each pair of adjacent images is input to a CNN that generates a feature vector fed into an LSTM to compute the embedding for each time step. Since the encoder is recurrent, it outputs a sequence of embeddings, where embedding z_i encodes the observed behavior from o_1 to o_{i+1}. The complete trajectory is encoded into z_{T-1} (i.e., z_D).

Since encoding and execution are only coupled by the embedding, the whole CBE network can be trained end-to-end. Through end-to-end learning, the encoder learns a behavior manifold that encodes path geometry (Sec.V). The embedding can be extremely low-dimensional (e.g., 32), which significantly saves memory compared to SLAM [13] or other learning-based approaches [12], [11].

C. Behavior-Conditioned Waypoint Generator

The waypoint generator (right side of Figure 2) executes a behavior while tracking the robot's progress along the

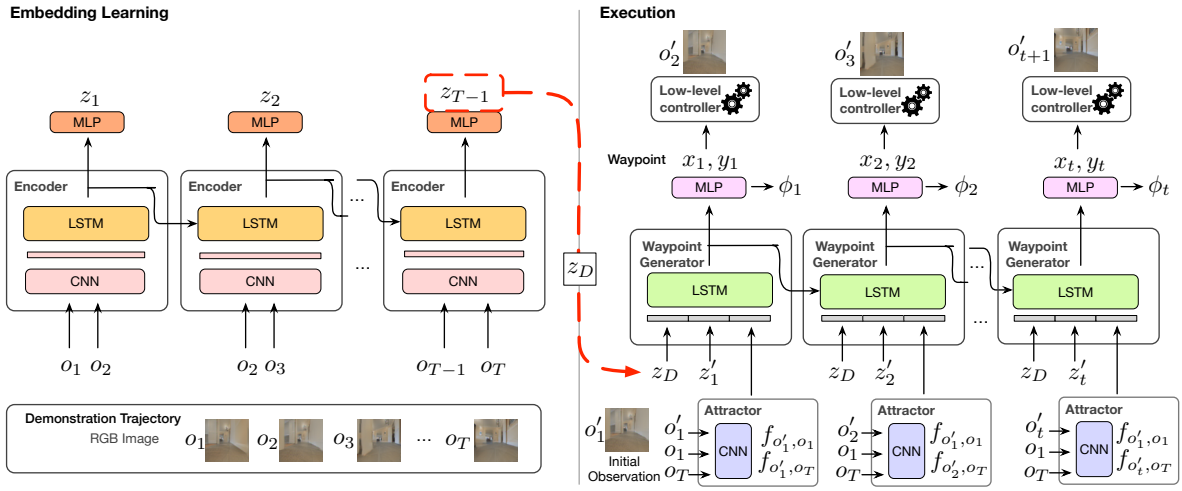


Fig. 2: Overview of CBE. The encoder compresses the image sequence $o_1, ..., o_T$ observed during a demonstration into a low dimensional embedding z_D. This is done via a recurrent LSTM network that inputs pairs of consecutive images, (o_t, o_{t+1}), and generates a sequence of latent embeddings, z_t, with the final z_{T-1} providing the overall embedding of the demonstration. In the execution phase, the waypoint generator uses the demonstration embedding, z_D, and the embedding of the images observed so far, z'_t, to generate the next waypoint, (x_t, y_t), and a measure of the progress thus far ϕ_t. The embeddings of the executed trajectory, z'_t, are computed using the same network as the demonstration encoding. An additional "Attractor" network processes the current image and the first and last images of the demonstration to provide information that helps with the alignment at the beginning and end of the trajectory. At each time step t, the waypoint is sent to the local controller, which moves the robot and provides the image for the next iteration, o'_{t+1}. This process is repeated until the robot reaches the goal o_T, indicated by $\phi = 1$.

demonstrated trajectory. The robot starts with its initial observation, o'_1, which does not have to exactly match the beginning of the demonstration, o_1. At every time step t, an LSTM unit takes as input the embedding z_D of the demonstration and the embedding z'_t of the images observed so far (computed using the same encoder network used for demonstration embeddings), along with features provided by an "Attractor" network described below. Using these, the recurrent unit predicts the next local waypoint, x_t, y_t, and the current progress, ϕ_t. The waypoint is input into the robot's low-level controller to generate motor commands. The low-level controller can be a simple PID controller, or it may support local obstacle avoidance [35]. The *progress indicator* ϕ_t provides the fraction of the demonstration the robot has completed at time t. It is used as a condition for behavior switching. After receiving the next observation, o'_{t+1}, new attractor features and embedding z'_{t+1} are computed and input to the next LSTM step. This process is repeated until the robot reaches the goal, indicated by $\phi = 1$.

Attractor Network. During execution, the robot's initial location and orientation may not be exactly the same as during demonstration, requiring the robot to align its initial location sufficiently well to follow the demonstrated trajectory. Similarly, to determine when the robot has reached the goal point, solely accumulating motion information from the observed images is not accurate enough. CBE solves these problems via the *attractor* network, which combines the robot's current observation with the initial and final demonstration observations (i.e, attractors) to provide features that can relate the current observation to the beginning and end of the demonstration. The attractor network is a CNN that generates $f_{o'_1,o_1}$ and $f_{o'_t,o_T}$ which are concatenated with the embedding (see Fig. 2 and 4).

D. Long Range Navigation via Behavior Segmentation

Since behavior embeddings are learned from egocentric observations, compounding error is inevitable, implying that z may not encode a complex long-horizon behavior precisely. We solve this by segmenting a long trajectory into a sequence of behaviors, each of which is specified by its embedding z_D and initial and final observations, o_1 and o_T, respectively. Via the attractor features, o_1 provides robustness toward noisy locations when starting a behavior, and o_T helps the behavior reach the goal location accurately enough to transition to the next behavior (related to funnels in LQR-Trees [36]).

We find fixed-distance segmentation works well in practice (Sec. V). Given an observation sequence $o_1, o_2, ...o_T$, we segment it into equally spaced segments, subject to the constraint that every segment contains no more than K observations, where K is determined by a validation set. Visual attractors are placed at the segmentation boundaries, and two adjacent segments share attractors.

Behavior Switching. When a robot executes a sequence of behaviors, it needs to know when it can safely switch from the current behavior to the next. It makes the switching decision by checking if the progress indicator of executing the current behavior ϕ_{current} is close to 1 (set to 0.95 in practice). If the condition holds, the robot resets its internal states and starts executing the next behavior z_{next}.

E. Composing Behaviors from Multiple Demonstrations

The segmentation method described in Sec. III-D can be extended to enable a robot to re-compose behavior segments from multiple demonstrations. In Figure 3, a robot is given demonstrations $A \to B$ and $C \to D$. If the attractor $\mathcal{A}_2$ and $\mathcal{A}'_2$ are close enough, then the robot can execute behaviors z_1, z'_2, z'_3 sequentially to go from A to D, even though

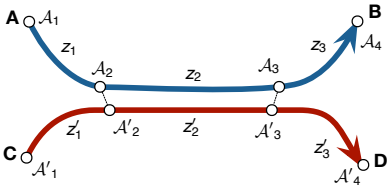


Fig. 3: Linking attractors from two demonstrations. $\mathcal{A}_i$, $\mathcal{A}'_j$ are attractors. z_i, z'_j are embeddings between attractors. Dotted lines are connections between attractors that are visually close.

no direct demonstration is available. This also allows us to further compress demonstrations by removing repeated behaviors (e.g., one of z_2 and z'_2).

Learning Choice Points. Fixed-distance segmentation does not guarantee that visual attractors from different demonstrations are placed at consistent locations, making it difficult to connect demonstrations. To mitigate this, we use a simple algorithm to find spatially consistent attractors. We train a classifier $d_t = C(o_{t-k+1}, ..., o_t)$ that takes the most recent k observations and predicts the next waypoint direction (discretized into 128 bins) using the training dataset. We compute the variance σ_t of the directional distribution to measure the uncertainty. Intuitively, o_t with high variance suggests that future trajectories may diverge and thus o_t is usually associated with spatially consistent locations such as intersections and doorways. Given a trajectory $o_1, ..., o_T$, we use C to compute the directional variances $\sigma_1, ..., \sigma_T$. Then we use a peak finding algorithm to find a set of choice points along a trajectory. While there are more sophisticated methods that can potentially find better choice points [37], we find this simple approach to be effective (see Sec. V-D).

IV. Implementation Details

We collected 100k trajectories from 18 large Gibson [38] environments as the training set. The trajectories are generated by a laser-based RMP controller [35] driving a non-holonomic car to follow a sequence of local waypoints computed by an A* planner. This controller also serves as the low-level controller for behavior execution. The low-level controller uses laser scans for local obstacle avoidance and in practice it could be replaced with vision-based controllers [17], [35] at extra computational cost. Simulation runs at 10 Hz. Image resolution is 64×64 with 120° field of view. Camera height is set to 1.0 m above the floor. All evaluations are conducted in 5 large unseen Gibson environments. These large environments are several times the size of an average Gibson environment, hence they are more suitable for evaluating long-horizon navigation performance.

We use a sequence length of 64 with a frame gap uniformly sampled between 0 to 2. Hence the average trajectory length is 128 time steps. We use the local waypoints in the same training set as supervision, and adopt DAgger [39] for data augmentation. In the DAgger phase, we jitter the robot's initial pose to simulate imperfect alignment and collect rollout trajectories generated by the current model. We then compute the correct waypoints and progress to train the next model. The correct local waypoints are computed by transforming (i.e., rotating and translating) the global ground truth waypoint associated with the closest trajectory sample to the robot. To compute the correct progress, we define the completed path as $o_1, ..., o_k$ where o_k is the closest observation to o'_t (in Euclidean distance). Hence ϕ_t is the fraction of completed path length to the total path length. By jittering the robot's pose in the DAgger phase, the robot learns a closed-loop policy that is robust to drift. This also enables the robot to robustly switch to the next behavior segment albeit the initial misalignment and errors in progress estimation by relating current observation to the attractors.

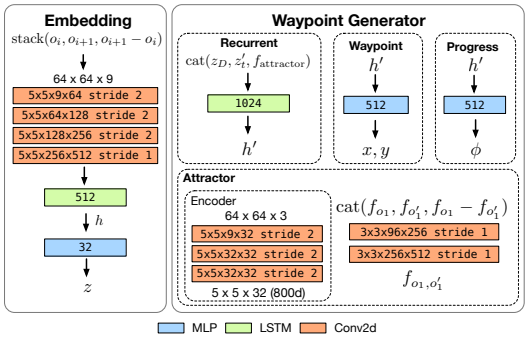


Fig. 4: Neural networks used by each component in CBE. See Figure 2 for how these components work together.

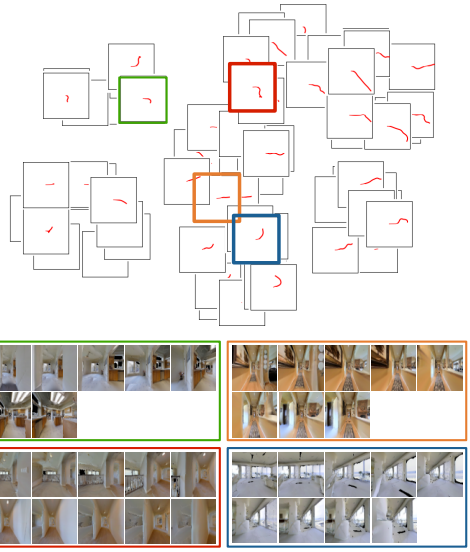


Fig. 5: t-SNE visualization of the behavior manifold. Each red line visualizes an encoded trajectory. The initial pose of the robot is always at the center, pointing rightwards. 4 example trajectories are shown at the bottom.

Network designs. Figure 4 details the network architectures of CBE modules. The networks are lightweight (70 MB) and can run in real time on an embedded system. We use the Adam optimizer with a learning rate of 0.0003 and a learning rate decay of 0.7. Every epoch contains 200k samples. We trained CBE for 5 epochs. All baselines were also trained using the same dataset for 5 to 7 epochs.

V. Experimental Results

A. Behavior Embedding

Figure 5 shows the t-SNE plot of embeddings extracted from training trajectories. The plot shows that the embedding space encodes a meaningful behavior manifold. From left to

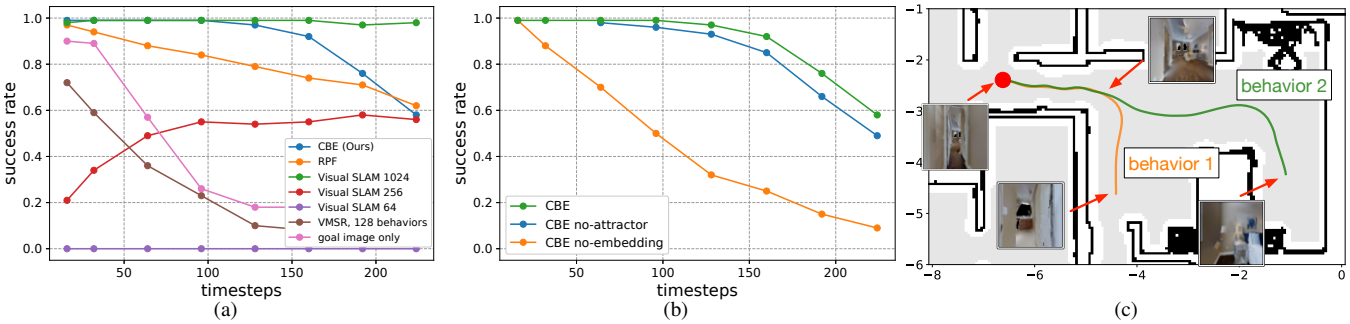


Fig. 6: Evaluating CBE for single-behavior navigation. (a) Comparing CBE with baselines. The abnormal degradation of Visual SLAM 256 for short trajectories is due to initialization failures. (b) Model ablation. (c) Example rollouts of two behaviors with similar structures.

right, trajectory lengths are increasing. From top to bottom, there is a smooth progression from "right turns" to "going straight" and to "left turns". The embedding space learns to encode visual odometry, even though it is not explicitly told to do so. We think this is why the learned embeddings generalize well to novel environments and can encode long-range behaviors, while being low-dimensional.

B. Single-behavior Navigation

We study how well a robot can navigate between two locations with a single CBE behavior in unseen environments. We collected a set of trajectories of lengths ranging from 16 time steps to more than 200 time steps, with 500 trajectories collected for each time step. We extract an embedding from each trajectory to condition the waypoint generator. We jitter the robot's initial pose to simulate imperfect alignment. We compare with the following baselines:

a) Visual SLAM: We adopt ORB-SLAM2 [13] which is one of the state-of-the-art real-time SLAM methods. We first feed the image sequence to reconstruct the environment and the trajectory. During execution, we run Visual SLAM in tracking mode which localizes and tracks the pose of the robot. We set the next waypoint to be the point on the trajectory that is 5 keyframes away from the robot's current location. If localization fails, the robot will use the previously computed waypoint until localization succeeds.

b) RPF: RPF [12] extracts a feature vector from each observation and uses attention to track the progress of a robot. Original RPF assumes the availability of camera pose and action at each time step. Here we only assume RPF has access to visual observations, same as ours.

c) VMSR: VMSR [8] clusters fixed-length demonstrations into a discrete set of behaviors. To support variable-length trajectories, we use a recurrent encoder similar to ours instead of a convolutional encoder. Again, VMSR uses raw observations as input.

d) Goal image only: we use the local controller in [11] because it shows strong performance when the goal image is visually reachable. We will compare [11] against CBE in Sec. V-C for its path following performance.

Figure 6a compares the success rates of CBE against the baselines (we also experimented with the SPL [40] metric with almost identical results). Using goal image alone shows poor performance due to visual occlusion. CBE achieves $> 95\%$ success rate for trajectories of up to 128 time steps

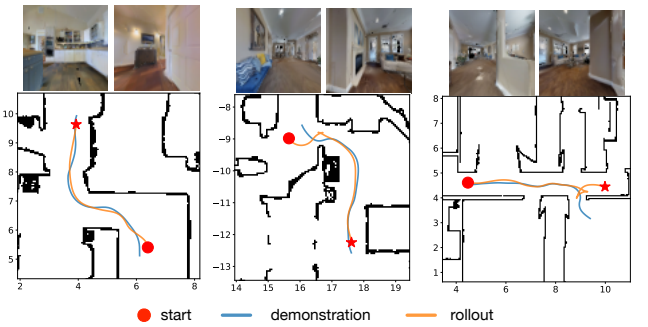


Fig. 7: Example traces (including one failure case) in test environments using a single behavior embedding. Start and goal images are shown at the top. Note that starting locations of the robot are not always aligned with the beginnings of the demonstrations.

(approx. 6 m in metric length). While CBE degrades for longer trajectories, we perform segmentation to maintain strong performance (Sec. V-C). RPF relies on accurate attention to track a path, but drift in attention may cause RPF to lose track and deviate from the path. VMSR clusters input trajectories into a discrete set of behaviors, hence it cannot capture the variations of behaviors well. While Visual SLAM outperforms single-embedding CBE for long trajectories, it degrades quickly as resolution decreases (degraded by 50% with 256×256 images and failed completely with 64×64 images). In contrast, CBE works well with low-resolution images, and can potentially be deployed on miniature robots with fast-moving cameras.

Figure 6c shows how learned embeddings can distinguish between two similar behaviors. These two behaviors share the same structure: go straight and turn right. However, behavior 2 needs to go straight for a longer distance before turning right. CBE captures the difference in distance so that a robot can reach both locations with no ambiguity. Figure 7 shows more example traces.

Model ablation. Figure 6b compares CBE with two variants. Removing the attractor model is detrimental because it would not be able to capture the initial misalignment. This effect is more pronounced when following a sequence of behaviors (Sec. V-C). Removing the embedding significantly degrades performance, as the robot has to rely on the goal attractor which can be occluded in long trajectories.

Robustness to unseen obstacles. To understand the robustness of our model in a dynamic environment, we randomly place a trashcan of size $0.3 \times 0.3 \times 1.0$ m close

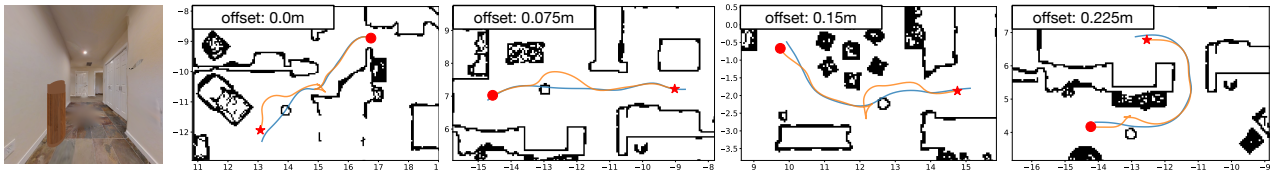
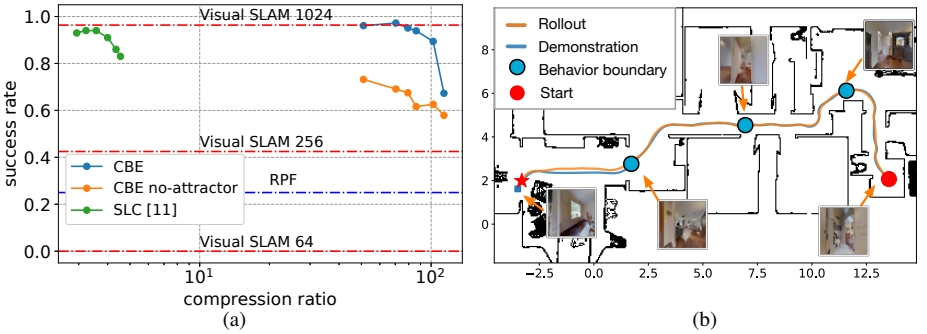


Fig. 8: Handling unseen obstacles during behavior execution. Left image: robot's view of the obstacle. 4 example executions with different obstacle offsets are shown on the right. Blue trajectory: demonstration. Orange trajectory: rollout. Red dot: starting location.



(a) (b)

Fig. 9: (a) Comparing success rates at different compression ratio. For Visual SLAM and RPF [12] we only show single success rates because they do not subsample observations. (b) CBE robustly follows a long path by segmenting the path into a sequence of behaviors. See the supplementary video for more examples.

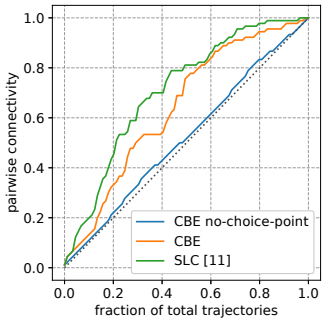


Fig. 10: Comparing map connectivity when only a fraction of total trajectories are used to build the maps. Diagonal line indicates no generalization.

to a trajectory and let the robot execute the corresponding behavior (128 time steps). Note that the behavior is encoded when the obstacle is not present. We evaluated our model on more than 300 trajectories and the following table shows the results by varying the offset of the obstacle to trajectories:

offset (m)	0.0	0.075	0.15	0.225
Success%	78.8	82.6	85.1	89.0

Figure. 8 shows example executions. The robot can successfully avoid most of the obstacles and reach the goal. The low-level controller *deliberately* makes the robot deviate from the demonstration to avoid the obstacle, but since CBE uses visual feedback to follow the encoded trajectory, it can generate corrective waypoints to get the robot back on track. Note that our model is trained without obstacles. Training the model with obstacles could further improve its robustness and we leave it as future work.

Robustness to actuation noise. We apply a random scale u to the controls of the robot at every time step. We first sample $x \sim \mathcal{N}(0.0, s/2)$ and then compute $u = \text{clip}(x, -s, s) + 1.0$. Intuitively, $s = 0.5$ means that we apply a +/- 50% random scale to velocity and steering angle (independently). We observed 2% and 8% degradation at $s = 0.5$ and 1.0, respectively. This shows that our model is robust to actuation noise.

C. Long-horizon Visual Path Following

A common navigation task that robots perform is to navigate between two places [17], [12], [11]. We show that by incorporating behaviors, a robot can follow a long trajectory with very sparse guidance. We sparsify a trajectory by segmenting it into a sequence of behaviors (Sec. III-D). We compare with [11] that sparsifies a trajectory by reasoning about target reachability. Compression ratio is defined as T/N, where N is the number of landmarks and T is the total number of observations. For CBE, N is approximately equal to the number of behaviors. We vary segment length K in Sec. III-D to adjust the number of behaviors. For Visual SLAM and RPF, we only report their success rates.

We selected semantically meaningful locations (e.g., rooms) in each test environment as starts and goals and generated 500 long trajectories with an average length of 20 m. A robot follows each trajectory with a jittered initial pose. Figure 9a compares the trajectory following success rates at different sparsity levels. Incorporating behaviors significantly increases sparsity compared to a behavior-less approach [11]. Without visual attractors, CBE performs considerably worse, as the robot is not able to calibrate its state well to switch to the next behavior reliably. Visual SLAM performs competitively with high-resolution images, but fails completely when using the same low-resolution images as other baselines. Figure 9b shows a qualitative example, where a 20 m long trajectory (450 time steps) is segmented into four behaviors and the robot executes the behaviors sequentially to reach the goal.

Memory efficiency. Table I shows that CBE is at least 10x more efficient at encoding visual demonstrations than the baselines. A trajectory sparsified by CBE usually contains fewer than 10 embeddings (32 floats each), interleaved by visual attractors (800 floats each). In comparison, Visual SLAM stores over ten thousand feature descriptors, and current learning-based methods require storing either dense visual features or raw images. This opens up opportunities to build compact topological maps of novel environments.

D. Behavior-based Topological Mapping

We follow the same setup as in [11], [18], where a robot builds a topological map of an environment from a set of experience trajectories consisting of RGB observations. A topological map is a directed graph where vertices are anchor observations selected from the trajectories and edges encode

Method	Avg. Mem (KB)	SR	Breakdown of memory usage per trajectory (average)
CBE (Ours)	**19**	**97.2**	6 attractors (3.2 KB each) + embeddings (32 floats = 128 bytes each)
SLAM [13] 1024×1024	341	96.3	10933 descriptors (32 bytes each)
256×256	199	42.5	6382 descriptors.
64×64	-	0.0	Failed to initialize.
RPF [12]	424	21.7	212 feature vectors (512 floats = 2 KB each)
SLC [11]	1548	93.7	129 images (12 KB each)

TABLE I: Comparing memory efficiency and success rate (SR) of different methods for long-horizon visual path following. All methods use 64×64 images except for SLAM which we evaluate on multiple resolutions. Note that SLAM requires extra memory to store the pose graph, which we did not include here due to the difficulty of estimating the value accurately.

Envs	**Calavo**			**Frierson**			**Kendall**			**Ooltewah**			**Sultan**		
#images	29,449			48,835			51,059			80,394			31,685		
	#verts	storage	SR	#verts	storage	SR	#verts	storage	SR	#verts	storage	SR	#verts	storage	SR
SPTM [18]	3067	6.0	3.3	5097	10.0	0.0	5320	10.4	0.0	8287	16.2	2.2	3290	6.4	0.0
SLC [11]	617	36.1	**97.8**	935	54.8	90.4	805	47.2	98.1	1115	65.3	**96.7**	759	44.5	86.7
CBE (Ours)	**357**	**1.2**	**97.8**	**498**	**1.6**	**100**	**490**	**1.6**	**100**	**611**	**2.0**	92.3	**388**	**1.3**	**98.9**

TABLE II: Comparing sizes of topological maps and planning success rates. Storage is in MegaBytes. SR indicates planning and trajectory following success rate. For SPTM we do a 10x subsampling of input observations. SLC does adaptive subsampling with a sparsification threshold of 0.98. For CBE we use $K = 100$ for creating behavior segments.

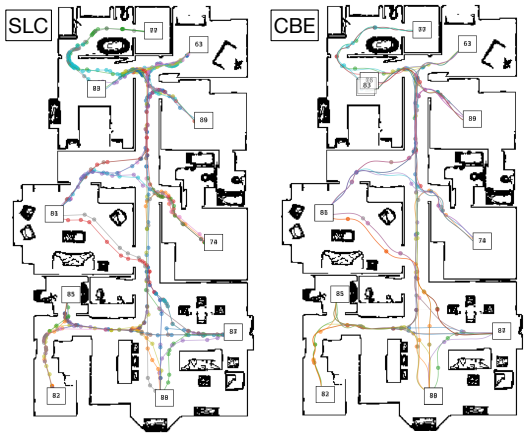


Fig. 11: Visualization of the topological maps built by SLC [11] (no behavior) and CBE (with behaviors) in one of the test environments (Calavo). Each circle is a vertex. Each demonstration is assigned a different color. See the supplementary video for more examples.

connectivity. This graph structure is often used in goal-conditioned navigation tasks, where a robot needs to plan a least-cost path to get to a specified goal.

We leverage behaviors to build sparse and well-connected topological maps. We first perform choice-point based segmentation as described in Sec. III-E, followed by distance-based segmentation (Sec. III-D) if needed. Each vertex stores an 800-dim attractor feature. Edges are either behavioral edges (via segmentation) or proximal edges (created by linking attractors). A robot first localizes itself and the goal using a network that predicts visual overlap [11]. Then the robot uses the Dijkstra algorithm to find the shortest path and executes the sequence of behaviors along the path.

We selected 10 to 14 semantically meaningful locations (e.g., rooms) in each test environment and collected pairwise trajectories to cover most of the traversable area. We built a topological map out of these trajectories. For planning, we let a robot start at one of the locations, plan a path to every other location, and follow the path. Table II compares the sizes of topological maps built by CBE and planning success rates against the baseline methods. CBE builds much more compact maps and is significantly more memory efficient. SPTM [18] subsamples observations and extracts a 512-dim feature vector from each observation. However, it is not robust enough for controlling a non-holonomic robot in a continuous state and action space. SLC [11] performs competitively, but at the expense of storing significantly more information per vertex (five 64×64 RGB images). The main failure cases of SLC are caused by faulty edges in the map due to visual aliasing. In comparison, CBE maps have much fewer vertices, store only an 800-dim feature per-vertex, and achieve similar or higher planning success rates due to less visual aliasing. Figure 11 visualizes the distribution of vertices in the map.

Impact of choice points on generalization. To see the necessity of using choice points for mapping, we evaluate pairwise connectivity between locations when using a fraction of all pairwise demonstrations to build the map. In Figure 10, we can see that without choice points there is almost no generalization. This is because fixed-distance segmentation (Sec. III-D) creates attractors that are inconsistently distributed in an environment, making it difficult to link attractors from different demonstrations. Detecting choice points significantly improves connectivity, but there is still a gap compared to a dense map. It can be a future work to improve choice point detection to close this gap.

VI. Conclusion

We introduce Composable Behavior Embedding, a robot-agnostic behavior representation for visual navigation. With CBE, robots are able to robustly replicate visual navigation tasks using extremely compact representations; two attractor features and a low-dimensional vector per behavior. We show how CBE can be incorporated into larger scale navigation systems for path following and topological mapping. Here, CBE significantly improves memory-efficiency. Our model operates in continuous state and action spaces, and we conducted experiments in realistic simulation environments. We will test our system on a real robot once the hardware becomes accessible, but based on other work using these environments we are confident that our results will transfer well to real environments and robots. The continuous trajectory embeddings learned by CBE are well suited to connect

to similarly structured language embeddings and using our model to perform language-based visual navigation is an interesting direction for future research.

VII. Acknowledgments

This work was funded in part by ONR grant 63-6094 and by the Honda Curious Minded Sponsored Research Agreement. We thank NVIDIA for generously providing a DGX used for this research via the NVIDIA Robotics Lab and the UW NVIDIA AI Lab (NVAIL).

References

[1] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta, "Discovering motor programs by recomposing demonstrations," in *International Conference on Learning Representations (ICLR)*, 2019.

[2] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy, "Task-conditioned variational autoencoders for learning movement primitives," in *Conference on Robot Learning (CoRL)*, 2019.

[3] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on Robot Learning (CoRL)*, 2019.

[4] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg, "Ddco: Discovery of deep continuous options for robot learning from demonstrations," in *Conference on Robot Learning (CoRL)*, 2017.

[5] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vazquez, and S. Savarese, "A behavioral approach to visual navigation with graph localization networks," *Robotics Science and Systems (RSS)*, 2019.

[6] N. Trigoni, A. Markham, and L. Xie, "SnapNav: learning mapless visual navigation with sparse directional guidance and visual reference," in *IEEE International Conference on Robotics and Automaton (ICRA)*, 2020.

[7] J. Roh, C. Paxton, A. Pronobis, A. Farhadi, and D. Fox, "Conditional driving from natural language instructions," in *Conference on Robot Learning (CoRL)*, 2019.

[8] A. Kumar, S. Gupta, and J. Malik, "Learning navigation subroutines from egocentric videos," in *Conference on Robot Learning (CoRL)*, 2019.

[9] T. Swedish and R. Raskar, "Deep visual teach and repeat on path networks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[10] N. Gopalan, E. Rosen, G. Konidaris, and S. Tellex, "Simultaneously learning transferable symbols and language groundings from perceptual data for instruction following," *Robotics Science and Systems (RSS)*, 2020.

[11] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *IEEE International Conference on Robotics and Automaton (ICRA)*, 2020.

[12] A. Kumar, S. Gupta, D. Fouhey, S. Levine, and J. Malik, "Visual memory for robust path following," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[14] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, September 2005, iSBN 0-262-20162-3.

[16] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[17] N. Hirose, F. Xia, R. Martín-Martín, A. Sadeghian, and S. Savarese, "Deep visual mpc-policy learning for navigation," *IEEE Robotics and Automation Letters*, 2019.

[18] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *International Conference on Learning Representations (ICLR)*, 2018.

[19] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese, "Scene memory transformer for embodied agents in long-horizon tasks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 538–547.

[22] A. Wahid, A. Toshev, M. Fiser, and T. E. Lee, "Long range neural navigation policies for the real world," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[23] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[24] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. Fox, "Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[25] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *Robotics Science and Systems (RSS)*, 2020.

[26] J. D. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings," in *International Conference on Machine Learning (ICML)*, 2018.

[27] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," in *International Conference on Learning Representations (ICLR)*, 2019.

[28] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia, "Compile: Compositional imitation learning and execution," in *International Conference on Machine Learning (ICML)*, 2019.

[29] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Learning movement primitive libraries through probabilistic segmentation," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 8, pp. 879–894, 2017.

[30] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *International Conference on Learning Representations (ICLR)*, 2019.

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

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

[33] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Neural autonomous navigation with riemannian motion policy," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[36] R. Tedrake, I. Manchester, M. Tobenkin, and J. Roberts, "Lqr-trees-feedback motion planning via sums of squares optimization," *Journal of Robotics Research (IJRR)*, vol. 29, pp. 1038–1052, 2010.

[37] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, 2020.

[38] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robotics and Automation Letters*, 2020.

[39] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[40] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.