A. Experimental details

The models used in all experiments are implemented in Tensorflow (Abadi et al., 2015) and use the Adam optimizer (Kingma & Ba, 2014).

A.1. Image navigation experiment

The training and test data sets are procedurally generated by sampling a random trajectory in randomly chosen images from the CelebA data set. The actions at each time steps are one-hot encoded (vector of size 5). The memorization phase is 256 time steps (we experimentally determined that this suffices to ensure that the agent usually reaches most positions in the environment), while the prediction one has 32 time steps during training and 256 during testing.

The VAE prior used in this experiment is obtained by creating a mixture distribution from the sufficient statistics of the frame encodings retrieved from the DND memory, whose weights are inversely proportional to the squared distances $d^{(k)}$ between $\mathbf{s}_t$ and the retrieved elements $\mathbf{s}_k^{(k)}$:

$$p_\theta(\mathbf{z}_t|\mathbf{s}_t, \mathbf{m}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{z}_t|\mathbf{z}^{(k)}) \; ; \quad w_k \propto \frac{1}{{d^{(k)}}^2 + \delta}$$

$\delta = 10^{-4}$ is added for numerical stability (Pritzel et al., 2017). In the DND memory we store sufficient statistics, but in this experiment we use the Euclidean distance between means in the nearest-neighbor search. (Alternatively, we could use the KL divergence between the distributions).

The VAE decoder and encoder use a 3-layered convolutional architecture to parameterize mean and variance of 16-dimensional latent states, but we noticed in practice that for this experiment even standard fully connected architectures perform well. In the transition model the standard deviation of the model is $r = 10^{-3}$. In the DND we retrieve the 5 nearest neighbour and use Euclidean distances between means.

The initial learning rate is 10^{-3}, and we anneal it linearly to $5 \cdot 10^{-5}$ during the first 50000 updates.

A.2. Labyrinth experiments

The data sets used for the labyrinth experiments contain 120000 action-conditioned videos, of which we use 100000 for training and 20000 for testing. Each video for the rotating agent experiments contains 80 frames. To form a training sequence we select randomly 49 consecutive frames of a video, that we split in 33 frames for the memorization phase and 16 for the prediction one. During testing, the prediction phase has 45 time steps. For the walking agent experiment the videos are 300 time steps. Similarly to the rotation experiment, to form a training sequence we get consecutive sequences of 150+32 time steps (memorization and prediction phase respectively). During testing, we use 150 frames for the prediction phase.

The transition noise of the SSM has standard deviation $r = 10^{-2}$. The pre-trained agent does not hit walls, therefore we do not need to handle non-linearities as in (4). To compute distances in the DND we map the state vectors to

$$\widetilde{\mathbf{s}}_t = \begin{bmatrix} \mathbf{s}_t^{(1)} \\ \mathbf{s}_t^{(2)} \\ \cos(\mathbf{s}_t^{(3)}) \\ \sin(\mathbf{s}_t^{(3)}) \end{bmatrix},$$

and optionally pass the resulting vector through a linear layer. This gives a 5-dimensional vector in a learned manifold in which we use the Euclidean distance (in our experiments, the model performed well even without the linear layer). In the DND we retrieve the 4 nearest neighbours.

In the VAE, we use convolutional encoder and decoder, and 64-dimensional latent state. The VAE prior $p_\theta(\mathbf{z}_t|\mathbf{s}_t, \mathbf{m})$ used in the Labyrinth experiments is slightly more involved than the mixture prior used in the image navigation one. It is formed using a *Generative Query Network* (Eslami et al., 2018), that first maps the data retrieved from memory $\{\mathbf{s}_i, \mathbf{z}_i\}$ with a MLP to an embedding vector h_t, and then combines the embedding h_t with the current state $\mathbf{s}_t$, mapping the result to the mean and variance of the Gaussian prior

$$h_t = f(\{\mathbf{s}_i, \mathbf{z}_i\})$$
$$p_\theta(\mathbf{z}_t|\mathbf{s}_t, \mathbf{m}) = \mathcal{N}(\boldsymbol{\mu}(h_t, \mathbf{s}_t), \boldsymbol{\sigma}(h_t, \mathbf{s}_t)).$$

The initial learning rate is set to $3 \cdot 10^{-3}$, and we linearly anneal it to $5 \cdot 10^{-5}$ during the first 100000 updates.

B. Videos of long-term generation

Videos of long-term generation from the GTM-SM for all the experiments of this paper are available at this Google Drive link (goo.gl/RXQPTL). The videos are subdivided in folders:

1. `videos/image_navigation/` contains the videos for the experiments in Section 4.1.
2. `videos/labyrinth_rotation/` contains the videos for the experiments in Section 4.2.1.
3. `videos/labyrinth_walk/` contains the videos for the experiments in Section 4.2.2.
4. `videos/labyrinth_walk_multirooms/` contains the videos for the experiments in Appendix C.

In all folders, the first video corresponds to the test sequence used to produce the figures in the paper.

C. Walking agent in Labyrinth (multiple rooms)

We consider the same trained model used for the results in section 4.2.2. In section 4.2.2, this model was tested on videos of length $T = 300$ of an agent walking in a single room, with both memorization and prediction phase of 150 time steps. To asses the long-term memorization and localization capabilities of the GTM-SM, we now test it on videos of the same agent walking in larger environments with multiple rooms. Each video is $T = 450$ time steps; we store 150 time steps in memory and we predict for 300 more. As the model is trained on single rooms, we cannot expect the VAE to correctly generate the corridors between rooms, but we can expect the model to be able to know its position and the textures in the room (i.e. the color of the walls and of the floor).

In Figure 4 we show the predictions from the model after more than 250 time steps from the end of the memorization phase. As expected, the model fails in drawing the walls that form the corridor between the two rooms. However, we see that the GTM-SM correctly remembers the texture of rooms that it has previously visited and is able to predict the change in the color of the floor in the corridor. This is better viewed looking at the videos of this experiment, available in the folder `videos/labyrinth_walk_multirooms/` in the supplementary material.

D. Inference network using landmark information

We now introduce an alternative inference network that uses the information in the DND memory to improve inference in cases in which the SSM transition model is not powerful enough to infer the correct position of the agent. We factorize the variational approximation $q_\phi(\mathbf{z}, \mathbf{s}|\mathbf{x}, \mathbf{a}, \mathbf{v})$ as

$$\begin{aligned} q_\phi(\mathbf{z}, \mathbf{s}|\mathbf{x}, \mathbf{a}, \mathbf{v}) &= q_\phi(\mathbf{z}|\mathbf{x}) q_\phi(\mathbf{s}|\mathbf{z}, \mathbf{a}, \mathbf{v}) \\ &= \prod_{t=\tau+1}^{T} q_\phi(\mathbf{z}_t|\mathbf{x}_t) q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t, \mathbf{z}_t, \mathbf{m}) \ . \end{aligned}$$

A graphical representation of the inference network of the GTM-SM is shown in Figure 5. $q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ is an inference network that outputs the mean and variance of a Gaussian distribution, as typically done in VAEs. The structured variational approximation $q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t, \mathbf{z}_t, \mathbf{m})$ retains the temporal dependency among the state variables and exploits the information stored in the memory $\mathbf{m}$. We define this approximation to depend on:

1. The *prior belief* $p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t)$. If at time $t-1$ we were at a given position, this dependence captures the fact that at time t we cannot be too far from it. This is the same dependence we used in Section 3.2.
2. *Landmark information*, obtained by querying the DND memory in the reverse direction with respect to the VAE prior, i.e., considering the frame encodings $\mathbf{z}_i$ as keys and the states $\mathbf{s}_i$ as values. At each time step the agent can then check whether it has already seen the current frame encoding $\mathbf{z}_t$ in the past, and exploit this information when computing the inferred position of the agent. We use $\mathbf{z}_t$ to query the reversed-DND, retrieving triplets $\{(\delta^{(k)}, \mathbf{z}^{(k)}, \mathbf{s}^{(k)}), k = 1, .., K'\}$ that are used in the computation of the parameters of $q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t, \mathbf{z}_t, \mathbf{m})$. Here, $\delta_i^{(k)}$ represents a distance in $\mathbf{z}$-space.

We define $q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t, \mathbf{z}_t, \mathbf{m})$ to be a Gaussian density, whose mean $\boldsymbol{\mu}_q$ and variance $\boldsymbol{\sigma}_q^2$ are the outputs of a neural network that merges the sufficient statistics $\boldsymbol{\mu}_p$ and $\boldsymbol{\sigma}_p^2$ of the prior $p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t)$, and the ones of the states $\mathbf{s}_i^{(k)}$ retrieved using landmark information: $\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2, k = 1 : K'$. We assume that we stored in the DND the mean and the variance of Gaussian latent states. The posterior mean is obtained as

$$\boldsymbol{\mu}_q = \boldsymbol{\mu}_p + \sum_{k=1}^{K'} \beta_k (\boldsymbol{\mu}_k - \boldsymbol{\mu}_p) \ ,$$

where $\beta_k \in [0, 1]$ is the output of a simple neural network with input $\delta_i^{(k)}$. The inference network can then learn to assign a high value to β_k whenever the distance in $\mathbf{z}$-space is small (i.e. the current observation is similar to a frame stored in the DND), so that the prior mean is moved in the direction of $\boldsymbol{\mu}_k$. Similarly, the posterior variance can be computed starting from the prior variance using another neural network: $\log \boldsymbol{\sigma}_q^2 = \log \boldsymbol{\sigma}_p^2 + NN(\delta_i^{1:K'}, \boldsymbol{\sigma}_{1:K'}^2, \boldsymbol{\sigma}_p^2)$.

With this choice for the inference network, the ELBO of the GTM-SM becomes

$$\begin{aligned} \mathcal{F}(\theta, \phi) = &\sum_{t=\tau+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_t)} \left[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t)\right] + \\ &- \mathbb{E}_{q_\phi^\star(\mathbf{s}_t)} \left[KL[q_\phi(\mathbf{z}_t|\mathbf{x}_t)||p_\theta(\mathbf{z}_t|\mathbf{s}_t, \mathbf{m})]\right] + \\ &- \mathbb{E}_{q_\phi^\star(\mathbf{s}_{t-1})} \left[KL\left[q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t, \mathbf{z}_t, \mathbf{m})||p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t)\right]\right] \ . \end{aligned}$$

with

$$q_\phi^\star(\mathbf{s}_t) = \int q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_t, \mathbf{z}_t, \mathbf{m}_{1:t-1}) q_\phi^\star(\mathbf{s}_{t-1}) \, \mathrm{d}\mathbf{s}_{t-1} \ .$$

Notice in particular the additional KL term for the SSM.

D.1. Image navigation with obstacles

We extend the image navigation experiments of Section 4.1 adding obstacles to the environment as illustrated in Figure 6 (left). We use displacement information as in

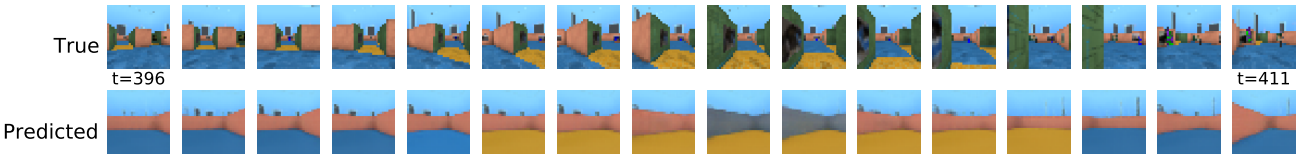

Figure 1: Prediction phase for a walking agent trained on one room and tested on multiple rooms. Time steps $t_{396:411}$.

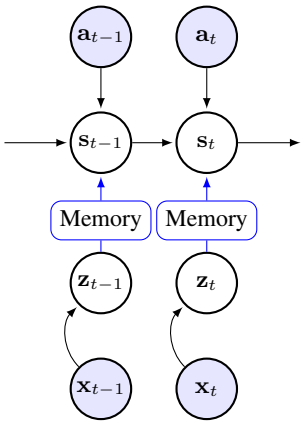

Figure 2: Inference network for the GTM-SM using landmark information. Blue arrows represent dependencies on the reversed DND memory.

the Labyrinth experiment of Section 4.2.2. The obstacles appear in random positions in each sequence, therefore we cannot learn a prior transition model that captures these non-linear dynamics. However, when doing inference the model can use its knowledge on the current frame $\mathbf{x}_t$ (that is not available during the prediction phase) to infer its position by exploiting landmark information.

To illustrate this we can look at the example in Figure 6 (right). At time $t-1$, the position $\mathbf{s}_{t-1}$ of the agent coincides with the red star. The agent's position together with the corresponding observation $\mathbf{x}_{t-1}$ (the yellow square) will be inserted in the DND memory. At time t, the agent receives a "move left" action; the prior transition probabilities will then predict that the agent has to move to the left (the green hexagon). Due to the presence of the obstacle however, the agent does not move, meaning that $\mathbf{x}_t$ will be the same as $\mathbf{x}_{t-1}$. Querying the DND in the reverse direction the model will then know that the inferred state (the blue dot) should be the same as the position at the previous time step that was stored in the DND (the red star).

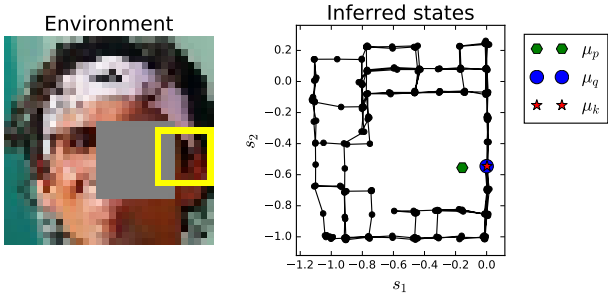

Figure 3: Image navigation experiment with obstacles. The agent (yellow square) cannot cross the ostacle (the gray squared area).