Incremental Scene Synthesis

Benjamin Planche[1,2] **Xuejian Rong**[3,4] **Ziyan Wu**[4] **Srikrishna Karanam**[4]
Harald Kosch[2] **YingLi Tian**[3] **Jan Ernst**[4] **Andreas Hutter**[1]

[1]Siemens Corporate Technology, Munich, Germany
[2]University of Passau, Passau, Germany
[3]The City College, City University of New York, New York NY
[4]Siemens Corporate Technology, Princeton NJ
{first.last}@siemens.com, {xrong,ytian}@ccny.cuny.edu, harald.kosch@uni-passau.de

Abstract

We present a method to incrementally generate complete 2D or 3D scenes with the following properties: (a) it is globally consistent at each step according to a learned scene prior, (b) real observations of a scene can be incorporated while observing global consistency, (c) unobserved regions can be hallucinated locally in consistence with previous observations, hallucinations and global priors, and (d) hallucinations are statistical in nature, *i.e.*, different scenes can be generated from the same observations. To achieve this, we model the virtual scene, where the active agent at each step can either perceive an observed part of the scene or generate a local hallucination. The latter can be interpreted as the agent's expectation at this step through the scene and can be applied, *e.g.*, to autonomous navigation. In the limit of observing real data at each point, our method converges to solving the SLAM problem. It can otherwise sample entirely imagined scenes from prior distributions. Besides autonomous agents, applications include problems where large data is required for building robust real-world applications, but few samples are available. We demonstrate efficacy on various 2D as well as 3D data.

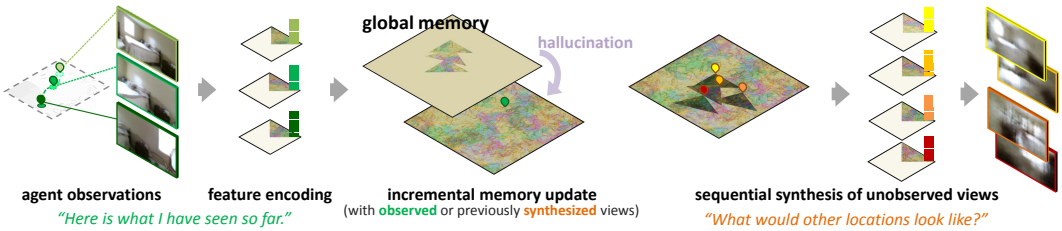

Figure 1: **Our solution** for scene understanding and novel view synthesis, given non-localized agents.

1 Introduction

We live in a three-dimensional world, and a proper cognitive understanding of its structure is crucial for acting and planning. The ability to anticipate under uncertainty is necessary for autonomous agents to perform various downstream tasks such as exploration and target navigation [4]. Deep learning has shown promise in addressing these questions [48, 26]. Given a set of views and corresponding camera poses, methods are able to learn an object's 3D shape via direct 3D or 2D supervision.

Existing *novel view synthesis* methods of this type have three common limitations. First, most recent approaches solely focus on single objects and surrounding viewpoints, and are trained with

Preprint. Under review.

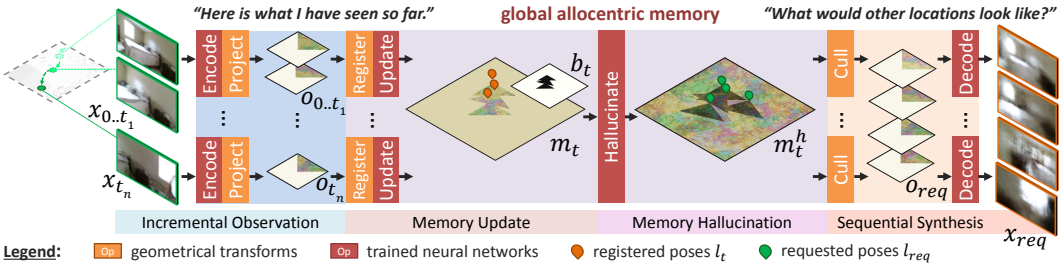

Figure 2: **Proposed pipeline** for non-localized agents exploring new scenes. Observations x_t are sequentially encoded and registered in a global feature map m_t with spatial properties, used to extrapolate unobserved content and generate consistent novel views x_{req} from requested viewpoints.

category-dependent 3D shape representations (*e.g.*, voxel, mesh, point cloud model) and 3D/2D supervision (*e.g.*, reprojection loss), which are not trivial to set up for natural scenes. While recent works on auto-regressive pixel generation [36], appearance flow prediction [48], or a combination of both [35] generate encouraging preliminary results for scenes, they only evaluate on data with mostly forwarding translation (*e.g.*, KITTI dataset [11]), and no scene understanding capabilities are convincingly shown. Second, these approaches assume that the camera poses are known precisely for all provided observations. This is a practically and biologically unrealistic assumption; an agent typically only has access to its own observations, not its precise location relative to objects in the scene (albeit it is provided by some oracle in synthetic environments, *e.g.*, [8]). Third, there are no constraints to guarantee consistency among the synthesized results.

In this paper, we address these issues with a unified framework that incrementally generates complete 2D or 3D scenes (*c.f.* Figure 1). Our solution builds upon the MapNet system [16], which offers an elegant solution to the registration problem but has no memory-reading capability. In comparison, our method not only provides a completely functional memory system, but also displays superior generation performance when compared to parallel deep reinforcement learning methods (*e.g.*, [10]). To the best of our knowledge, our solution is the first complete end-to-end trainable read/write allocentric spatial memory for visual inputs. Our key contributions are summarized below:

- Starting with only scene observations from a non-localized agent (*i.e.*, no location/action inputs unlike, *e.g.*, [10]), we present novel mechanisms to update a global memory with encoded features, hallucinate unobserved regions and query the memory for novel view synthesis.
- Memory updates are done with either observed or hallucinated data. Our domain-aware mechanism is the first to explicitly ensure the representation's global consistency w.r.t. the underlying scene properties in both cases.
- We propose the first framework that integrates observation, localization, globally consistent scene learning, and hallucination-aware representation updating to enable incremental scene synthesis.

We demonstrate the efficacy of our system on a variety of 2D partially observable synthetic and realistic environments. Finally, to establish scalability, we also evaluate the proposed model on challenging 3D environments.

2 Related Work

Our work is related to localization, mapping, and novel view synthesis. We discuss relevant work to provide some context.

Neural Localization and Mapping. The ability to build a global representation of an environment, by registering frames captured from different viewpoints, is key to several concepts such as reinforcement learning or scene reconstruction. Recurrent neural networks are commonly used to accumulate features from image sequences, *e.g.* to predict the camera trajectory [25, 32]. Extending these solutions with a queryable memory, state-of-the-art models are mostly egocentric and action-conditioned [4, 29, 46, 10, 24]. Some oracle is, therefore, usually required to provide the agent's action at each time step t [24]. This information is typically used to regress the agent state s_t *e.g.*,

its pose, which can be used in a memory structure to index the corresponding observation x_t or its features. In comparison, our method solely relies on the observations to regress the agent's pose.

Progress has also been made towards solving visual SLAM with neural networks. CNN-SLAM [37] replaced some modules in classical SLAM methods [7] with neural components. Neural SLAM [46] and MapNet [16] both proposed a spatial memory system for autonomous agents. Whereas the former deeply interconnects memory operations with other predictions (*e.g.*, motion planning), the latter offers a more generic solution with no assumption on the agents' range of action or goal. Extending MapNet, our proposed model not only attempts to build a map of the environment, but also makes incremental predictions and hallucinations based on both past experiences and current observations. This capability of predicting under uncertainty is critical in many scenarios.

3D Modeling and Geometry-based View Synthesis. Much effort has also been expended to explicitly modeling the underlying 3D structure of scenes and objects, *e.g.* [7, 6]. While appealing and accurate results are guaranteed when multiple source images are provided, this line of work is fundamentally not able to deal with sparse inputs. To address this issue, Flynn *et al.* [9] proposed a deep learning approach focused on the multi-view stereo problem by regressing directly to output pixel values. On the other hand, Ji *et al.* [18] explicitly utilized learned dense correspondences to predict the image in the middle view of two source images. Generally, these methods are limited to synthesizing a middle view among fixed source images, whereas our framework is able to generate arbitrary target views by extrapolating from prior domain knowledge.

Novel View Synthesis. The problem we tackle here can be formulated as a novel view synthesis task: given pictures taken from certain poses, solutions need to synthesize an image from a new pose, and has seen significant interest in both vision [26, 48] and graphics [15]. There are two main flavors of novel view synthesis methods. The first type synthesizes pixels from an input image and a pose change with an encoder-decoder structure [36]. The second type reuses pixels from an input image with a sampling mechanism. For instance, Zhou *et al.* [48] recasted the task of novel view synthesis as predicting dense flow fields that map the pixels in the source view to the target view, but their method is not able to hallucinate pixels missing from source view. Recently, methods that use geometry information have gained popularity, as they are more robust to large view changes and resulting occlusions [26]. However, these conditional generative models rely on additional data to perform their target tasks. In contrast, our proposed model enables the agent to predict its own pose and synthesize novel views in an end-to-end fashion.

3 Methodology

While the current state of the art in scene registration yields satisfying results, there are several assumptions, including prior knowledge of the agent's range of actions, as well as the actions a_t themselves at each time step. In this paper, we consider unknown agents, with only their observations x_t provided during the memorization phase. In the spirit of the MapNet solution [16], we use an allocentric spatial memory map. Projected features from the input observations are registered together in a coordinate system relative to the first inputs, allowing to regress the position and orientation (*i.e.*, *pose*) of the agent in this coordinate system at each step. Moreover, given viewpoints and camera intrinsic parameters, features can be extracted from the spatial memory (*frustum culling*) to recover views. Crucially, at each step, memory "holes" can be temporarily filled by a network trained to generate domain-relevant features while ensuring global consistency. Put together (*c.f.* Figure 2), our pipeline (trainable both separately and end-to-end) can be seen as an explicit topographic memory system with localization, registration, and retrieval properties, as well as consistent memory-extrapolation from prior knowledge. We detail our proposed approach in this section.

3.1 Localization and Memorization

Our solution first takes a sequence of observed images $x_t \in \mathbb{R}^{c \times h \times w}$ (*e.g.*, with $c = 3$ for RGB images or 4 for RGB-D ones) for $t = 1, \ldots, \tau$ as input, localizing them and updating the spatial memory $m \in \mathbb{R}^{n \times u \times v}$ accordingly. The memory m is a discrete global map of dimensions $u \times v$ and feature size n. m_t represents its state at time t, after updating m_{t-1} with features from x_t.

Encoding Memories. Observations are encoded to fit the memory format. For each observation, a feature map $x'_t \in \mathbb{R}^{n \times h' \times w'}$ is extracted by an encoding convolutional neural network (CNN). Each

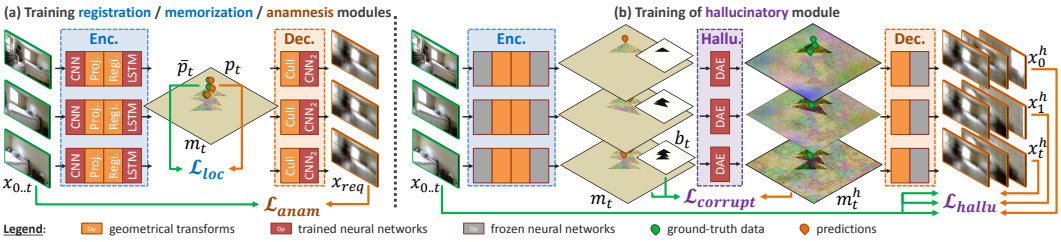

Figure 3: **Pipeline training**. Though steps are shown separately in the figure (for clarity), the method is trained in a single pass.

feature map is then projected from the 2D image domain into a tensor $o_t \in \mathbb{R}^{n \times s \times s}$ representing the agent's spatial neighborhood (to simplify later equations, we assume u, v, s are odd). This operation is data and use-case dependent. For instance, for RGB-D observations (or RGB images extended by some monocular depth estimation method, *e.g.*, [44]), the feature maps are first converted into point clouds using the depth values and the camera intrinsic parameters (assuming like Henriques and Vedaldi [16] that the ground plane is approximately known). They are then projected into o_t through discretization and max-pooling (to handle many-to-one feature aggregation, *i.e.*, when multiple features are projected into the same cell [30]). For 2D scenes (*i.e.*, agents walking on an image plane), o_t can be directly obtained from x_t (with optional cropping/scaling).

Localizing and Storing Memories. Given a projected feature map o_t and the current memory state m_{t-1}, the registration process involves densely matching o_t with m_{t-1}, considering all possible positions and rotations. As explained by Henriques and Vedaldi [16], this can be efficiently done through cross-correlation. Considering a set of r yaw rotations, a bank $o'_t \in \mathbb{R}^{r \times n \times s \times s}$ is built by rotating o_t r times: $o'_t = \left\{ R(o_t\ ,\ 2\pi \frac{i}{r}\ ,\ c_{s,s}) \right\}_{i=0}^{r}$, with $c_{s,s} = (\frac{s+1}{2}, \frac{s+1}{2})$ horizontal center of the patch, and $R(o, \alpha, c)$ the function rotating each element in o around the position c by an angle α, in the horizontal plane. The dense matching can therefore be achieved by sliding this bank of r feature maps across the global memory m_{t-1} and comparing the correlation responses. In other terms, the localization probability field $p_t \in \mathbb{R}^{r \times u \times v}$ is efficiently obtained by computing the cross-correlation (*i.e.*, "*convolution*", operator $\star$, in deep learning literature) between m_{t-1} and o'_t and normalizing the response map (*softmax* activation σ). The higher a value in p_t, the stronger the belief the observation comes from the corresponding pose. Given this probability map, it is possible to register o_t into the global map space (*i.e.*, rotating and translating it according to p_t estimation) by directly convolving o_t with p_t. This registered feature tensor $\hat{o}_t \in \mathbb{R}^{n \times u \times v}$ can finally be inserted into memory:

$$m_t = \text{LSTM}(m_{t-1},\ \hat{o}_t,\ \theta_{lstm}) \quad \text{with} \quad \hat{o}_t = p_t * o'_t \quad \text{and} \quad p_t = \sigma(m_{t-1} \star o'_t) \tag{1}$$

A long short-term memory (LSTM) unit is used, to update m_{t-1} (the unit's *hidden* state) with $\hat{o}_t$ (the unit's input) in a knowledgeable manner (*c.f.* trainable parameters θ_{lstm}). During training, the recurrent network will indeed learn to properly blend overlapping features, and to use $\hat{o}_t$ to solve potential uncertainties in previous insertions (uncertainties in p result in blurred $\hat{o}$ after convolution). The LSTM is also trained to update an occupancy mask of the global memory, later used for constrained hallucination (*c.f.* Section 3.3).

Training. The aforementioned process is trained in a supervised manner given the ground-truth agent's poses. For each sequence, the feature vector $o_{t=0}$ from the first observation is registered at the center of the global map without rotation (origin of the allocentric system). Given $\bar{p}_t$ the one-hot encoding of the actual state at time t, the network's loss $\mathcal{L}_{loc}$ at time τ is computed over the remaining predicted poses using binary cross-entropy:

$$\mathcal{L}_{loc} = \frac{1}{\tau} \sum_{t=1}^{\tau} \big[-\bar{p}_t \cdot \log(p_t) + (1 - \bar{p}_t) \cdot \log(1 - p_t) \big] \tag{2}$$

3.2 Anamnesis

Applying a novel combination of geometrical transforms and decoding operations, memorized content can be recalled from m_t and new images from unexplored locations synthesized. This process can be seen as a many-to-one recurrent generative network, with image synthesis conditioned by the global

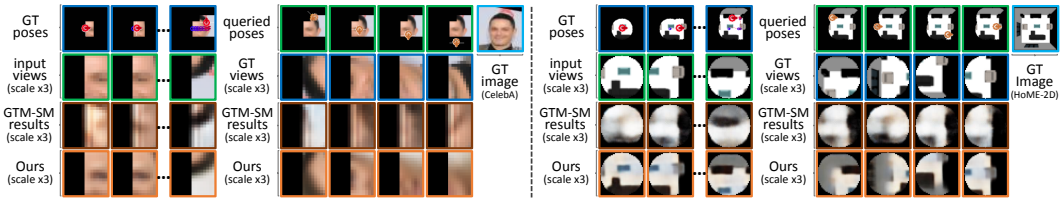

Figure 4: **Synthesis of memorized and novel views** from 2D scenes. Our method benefits from prior knowledge and global representation.

memory and the requested viewpoint. We present how the entire network can thus be advantageously trained as an auto-encoder with a recurrent neural encoder and a persistent latent space.

Culling Memories. While a decoder can retrieve observations conditioned by the full memory and requested pose, it would have to disentangle the visual and spatial information itself, which is not trivial to learn (*c.f.* ablation study in Section 4.1). Instead, we propose to use the spatial properties of our memory to first *cull* the features from requested viewing volumes, before passing them as only inputs to our decoder. More formally, given the allocentric coordinates $l_{req} = (u_{req}, v_{req})$, orientation $\alpha_{req} = 2\pi \frac{r_{req}}{r}$, and view field α_{fov}, $o_{req} \in \mathbb{R}^{n \times s \times s}$ representing the requested neighborhood is filled as follow:

$$o_{req,kij} = \begin{cases} \hat{o}_{req,kij} & \text{if } \operatorname{atan2} \frac{j - \frac{s+1}{2}}{i - \frac{s+1}{2}} < \frac{\alpha_{fov}}{2} \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

with $\hat{o}_{req}$ the unculled feature patch extracted from m_t rotated by $-\alpha_{req}$, *i.e.*, $\forall k \in [0\,..\,n-1]$, $\forall (i,j) \in [0\,..\,s-1]^2$:

$$\hat{o}_{req,kij} = R(m_t,\ -\alpha_{req},\ c_{u,v} + l_{req})_{k\xi\eta} \quad \text{with} \quad (\xi, \eta) = (i,j) + c_{u,v} + l_{req} - c_{s,s} \tag{4}$$

This differentiable operation combines feature extraction (through translation and rotation) and *viewing frustum culling* (*c.f.* computer graphics to render large 3D scenes).

Decoding Memories. As input observations undergo encoding and projection, feature maps culled from the memory go through a reverse procedure to be projected back into the image domain. With the synthesis conditioning covered in the previous step, a decoder directly takes o_{req} (*i.e.*, the view-encoding features) and returns x_{req}, the corresponding image. This back-projection is still a complex task. The decoder must both project the features from voxel domain to image plane, and decode them into visual stimuli. Previous works and qualitative results demonstrate that a well-defined (*e.g.*, *geometry-aware*) network can successfully accomplish this task.

Training. By requesting the pipeline to recall given observations—*i.e.*, setting $l_{req,t} = \bar{l}_t$ and $r_{req,t} = \bar{r}_t$, $\forall t \in [1, \tau]$, with $\bar{l}_t$ and $\bar{r}_t$ the agent's ground-truth position/orientation at each step t—it can be trained end-to-end as an image-sequence auto-encoder (*c.f.* Figure 3.a). Therefore, its loss $\mathcal{L}_{anam}$ is computed as the L1 distance between x_t and $x_{req,t}$, $\forall t \in [0, \tau]$, averaged over the sequences. Note that thanks to our framework's modularity, the global map and registration steps can be removed to pre-train the encoder and decoder together (passing the features directly from one to the other). We observe that such a pre-training tends to stabilize the overall learning process.

3.3 Mnemonic Hallucination

While the presented pipeline can generate novel views, these views have to overlap previous observations for the solution to extract enough features for anamnesis. Therefore, we extend our memory system with an *extrapolation* module to *hallucinate* relevant features for unexplored regions.

Hole Filling with Global Constraints. Under global constraints, we build an auto-encoder in the feature domain, which takes m_t as input and returns a convincingly hole-filled version m_t^h, while leaving registered features uncorrupted. In other words, this module should provide relevant features which seamlessly integrate existing content according to prior domain knowledge.

Training. Assuming the agent homogeneously explores training environments, the hallucinatory module is trained at each step $t \in [0, \tau_{cur}]$ by generating m_t^h, hole-filled memory used to predict

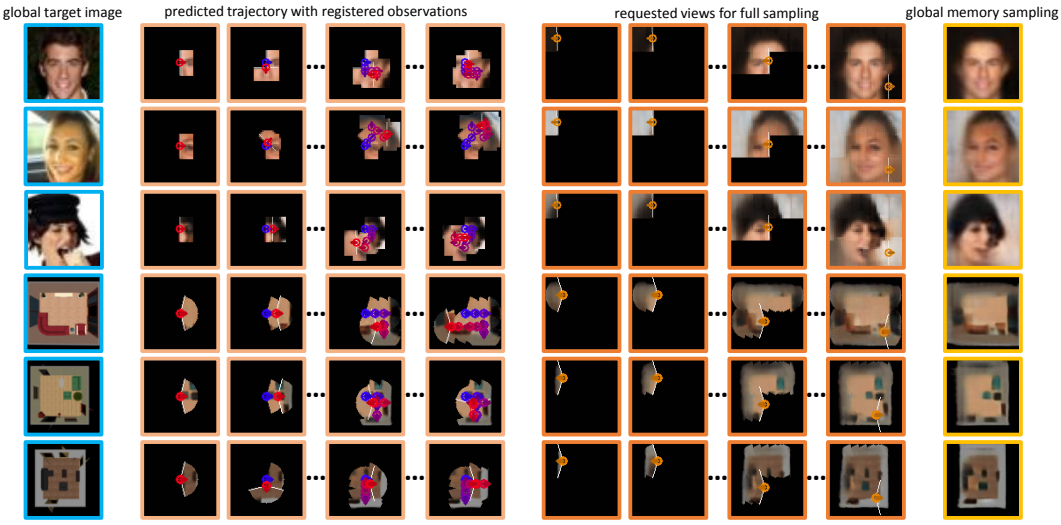

Figure 5: **Incremental and direct memory sampling** of complete environments from partial observations (on CelebA).

yet-to-be-observed views $\{x_t\}_{t=\tau_{cur}+1}^{\tau}$. To ensure that registered features are not corrupted, we also verify that all observations $\{x_t\}_{t=0}^{\tau_{cur}}$ can be retrieved from m_t^h (*c.f.* Figure 3.b). This generative loss is computed as follows:

$$\mathcal{L}_{hallu} = \frac{1}{\tau(\tau-1)} \sum_{t=0}^{\tau-1} \sum_{i=0}^{\tau} |x_i^h - x_i|_1 \tag{5}$$

with x_i^h the view recovered from m_i^h using the agent's true location $\bar{l}_i$ and orientation $\bar{r}_i$ for its observation x_i. Additionally, another loss is directly computed in the feature domain, using memory occupancy masks b_t to penalize any changes to the registered features (given $\odot$ Hadamard product):

$$\mathcal{L}_{corrupt} = \frac{1}{\tau} \sum_{t=0}^{\tau} |(m_t^h - m_t) \odot b_t|_1 \tag{6}$$

Trainable end-to-end, our model efficiently acquires domain knowledge to register, hallucinate, and synthesize scenes.

4 Experiments

We demonstrate our solution on various synthetic and real 2D and 3D environments. For each experiment, we consider an unknown agent exploring an environment, only providing a short sequence of partial observations (limited view field). Our method has to localize and register the observations, and build a global representation of the scene. Given a set of requested viewpoints, it should then render the corresponding views. In this section, we qualitatively and quantitatively evaluate the predicted trajectories and views, comparing with GTM-SM [10], the only other end-to-end memory system for scene synthesis, based on the Generative Query Network [8].

4.1 Navigation in 2D Images

We first study agents exploring images (randomly walking, accelerating, rotating), observing the image patch in their view field at each step (more details and results in the supplementary material).

Experimental Setup. We use a synthetic dataset of indoor 83×83 floor plans rendered using the HoME platform [2] and SUNCG data [34] (8,640 training + 2,240 test images from random rooms "*office*", "*living*", and "*bedroom*"). Similar to Fraccaro *et al.* [10], we also consider an agent exploring real pictures from the CelebA dataset [21], scaled to 43×43px. We consider two types of agents for each dataset. To reproduce Fraccaro *et al.* [10] experiments, we first consider non-rotating agents

Table 1: **Quantitative comparison 2D and 3D scenes**, *c.f.* setups in Subsections 4.1-4.2 (↘ *the lower the better;* ↗ *the higher the better; "u" horizontal bin unit according to AVD setup).*

Exp.	Methods	Average Position Error			Absolute Trajectory Error			Anam. Metr.		Hall. Metr.	
		Med.↘	Mean↘	Std.↘	Med.↘	Mean↘	Std.↘	L1↘	SSIM↗	L1↘	SSIM↗
A) A^s_{cel}	GTM-SM	4.0px	4.78px	4.32px	6.40px	6.86px	3.55px	0.14	0.57	0.14	0.41
	Ours	**1.0px**	**0.68px**	**1.02px**	**0.49px**	**0.60px**	**0.64px**	**0.06**	**0.80**	**0.09**	**0.72**
B) A^c_{cel}	GTM-SM	3.60px	5.04px	4.42px	2.74px	1.97px	2.48px	0.21	0.50	0.32	0.41
	Ours	**1.0px**	**2.21px**	**3.76px**	**1.44px**	**1.72px**	**2.25px**	**0.08**	**0.79**	**0.20**	**0.70**
C) A^s_{cel}	GTM-SM	4.0px	4.78px	4.32px	6.40px	6.86px	3.55px	0.14	0.57	0.14	0.41
	Ours	**1.0px**	**0.68px**	**1.02px**	**0.49px**	**0.60px**	**0.64px**	**0.06**	**0.80**	**0.09**	**0.72**
D) Doom	GTM-SM	1.41u	2.15u	**1.84u**	**1.73u**	**1.81u**	**1.06u**	0.09	0.52	0.13	0.49
	Ours	**1.00u**	**1.64u**	2.16u	1.75u	1.95u	1.24u	0.09	**0.56**	**0.11**	**0.54**
E) AVD	GTM-SM	1.00u	0.77u	0.69u	0.31u	0.36u	0.40u	0.37	0.12	0.43	0.10
	Ours	**0.37u**	**0.32u**	**0.26u**	**0.20u**	**0.21u**	**0.18u**	**0.22**	**0.31**	**0.25**	**0.23**

Table 2: **Ablation study** on CelebA with agent A^c_{cel}. Removed modules are replaced by identity mappings; remaining ones are adapted to the new input shapes when necessary. LSTM, memory, and decoder are present in all instances ("Localization" is the MapNet module).

Pipeline Modules				Anamnesis Metrics		Hallucination Metrics	
Encoder	Localization	Hallucinatory DAE	Culling	L1↘	SSIM↗	L1↘	SSIM↗
∅	∅	∅	∅	0.18	0.62	0.24	0.59
✓	∅	∅	∅	0.17	0.62	0.24	0.58
✓	✓	∅	∅	0.15	0.66	0.20	0.61
✓	✓	✓	∅	0.15	0.65	0.19	0.62
✓	∅	✓	✓	0.14	0.69	0.19	0.63
∅	✓	✓	✓	0.13	0.71	0.17	0.66
✓	✓	∅	✓	**0.08**	**0.80**	0.18	0.66
✓	✓	✓	✓	**0.08**	**0.80**	**0.15**	**0.70**

A^s—only able to translate in the 4 directions—with a $360°$ view field covering an image patch centered on the agents' position. The CelebA agent A^s_{cel} has a 15×15px square view field; while the view field of the HoME-2D agent A^s_{hom} reaches 20px away, and is therefore circular (in the 41×41 patches, pixels further than 20px are left blank). To consider more complex scenarios, agents A^c_{cel} and A^c_{hom} are also designed. They can rotate and translate (in the gaze direction), observing patches rotated accordingly. On CelebA images, A^c_{cel} can rotate by $\pm 45°$ or $\pm 90°$ each step, and only observes 8×15 patches in front ($180°$ rectangular view field); while for HoME-2D, A^c_{hom} can rotate by $\pm 90°$ and has a $150°$ view field limited to 20px. All agents can move from $1/4$ to $3/4$ of their view fiel each step. Input sequences are 10 steps long. For quantitative studies, methods have to render views covering the whole scenes w.r.t. the agents' properties.

Qualitative Results. As shown in Figure 4, our method efficiently uses prior knowledge to register observations and extrapolate new views, consistent with the global scene and requested viewpoints. While an encoding of the agent's actions is also provided to GTM-SM (guiding the localization), it cannot properly build a global representation from short input sequences, and thus fails at rendering completely novel views. Moreover, unlike the dictionary-like memory structure of GTM-SM, our method stores its representation into a single feature map, which can therefore be queried in several ways. As shown in Figure 5, for a varying number of conditioning inputs, one can request novel views one by one, culling and decoding features; with the option to register hallucinated views back into memory (*i.e.*, saving them as "valid" observations to be reused). But one can also directly query the full memory, training another decoder to convert all the features. Figure 7 also demonstrates how different trajectories may lead to different intermediate representations, although they converge as the scene coverage increases.

Quantitative Evaluations. We quantitatively evaluate the methods' ability to register observations at the proper positions in their respective coordinate systems (*i.e.*, to predict agent trajectories), to retrieve observations from memory, and to synthesize new ones. For localization, we measure the average position error (APE) and the absolute trajectory error (ATE), commonly used to evaluate SLAM systems [6]. For image synthesis, we make the distinction between recalling images already observed (*anamnesis*) and generating unseen views (*hallucination*). For both, we compute the

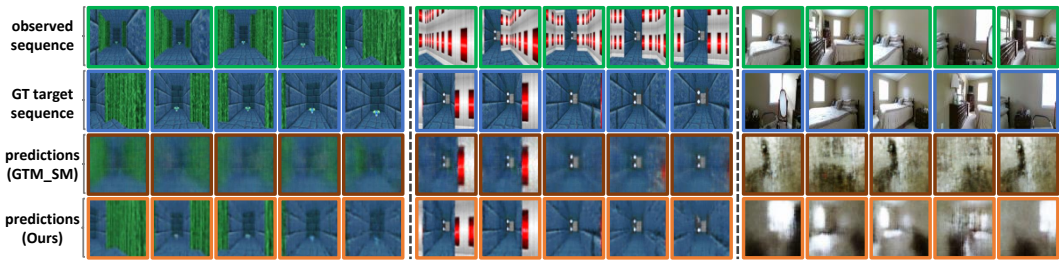

Figure 6: **Qualitative comparison on 3D use-cases**, w.r.t. anamnesis and hallucination.

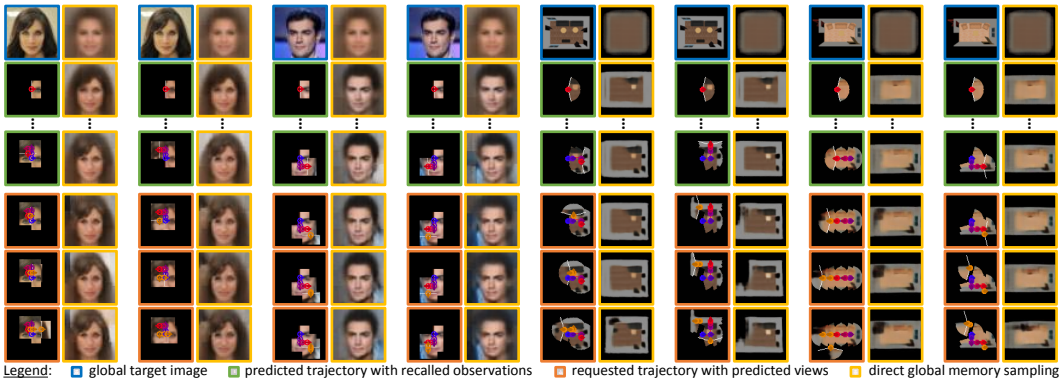

Figure 7: **Incremental exploration and hallucination** (on 2D use-cases). Scene representations evolve with the registration of observed or hallucinated views (*e.g.*, adapting hair color, face orientation, furniture, *etc.*).

common L1 distance between predicted and expected values, and the structural similarity (SSIM) index [40] for the assessment of perceptual quality [39, 45].

Table 1.A-C shows the comparison on 2D cases. Though GTM-SM leverages the provided actions to infer trajectories, our method is overall more precise, directly using the observations. Moreover, while GTM-SM fares well enough in recovering seen images from memory, it cannot synthesize views out of the observed domain. Our method not only extrapolates adequately from prior knowledge, but also generates views which are consistent from one to another (*c.f.* Figure 7 showing views stitched into a consistent global image). Note that on a Nvidia Titan X, the whole process (registering 5 views, localizing the agent, recalling the 5 images, and generating 5 new ones) takes less than 1s.

Ablation Study. Results of an ablation study are shown in Table 2 to further demonstrate the contribution of each module. Note that the APE/ATE are not represented, as they stay constant as long as the MapNet localization is included. In other words, our extensions cause no regression in terms of localization. Localizing and clipping features facilitate the decoding process by disentangling the visual and spatial information, thus improving the synthesis quality. Hallucinating features directly in the memory ensures image consistency.

4.2 Exploring Virtual and Real 3D Scenes

We finally demonstrate the capability of our method on the more complex case of 3D scenes.

Experimental Setup. As a first 3D experiment, we recorded, with the Vizdoom platform [43], 34 training and 6 testing episodes of 300 RGB-D observations from a human-controlled agent navigating in various static virtual scenes (walking with variable speed or rotating by 30° each step). Poses are discretized into 2D bins of 30×30 game units. Trajectories of 10 continuous frames are sampled and passed to the methods (the first 5 images as observations, and the last 5 as training ground-truths). We then consider the Active Vision Dataset (AVD) [1] which covers various real indoor scenes, often capturing several rooms per scene. We selected 15 for training and 4 for testing as suggested by the dataset authors, for a total of $\sim 20,000$ RGB-D images densely captured every 30cm (on a 2D grid) and every 30° in rotation. For each scene we randomly sampled $5,000$ agent trajectories of

10 frames each (each step the agent goes forward with 70% probability or rotates either way, to favor exploration). For both experiments, the 10-frame sequences are passed to the methods—the first 5 images as observations and the last 5 as ground-truths during training. Again, GTM-SM also receives the action encodings. For our method, we opted for $m \in \mathbb{R}^{32\times43\times43}$ for the Doom setup and $m \in \mathbb{R}^{32\times29\times29}$ for the AVD one.

Qualitative Results. Though a denser memory could be used for more refined results, Figure 6 shows that our solution is able to register meaningful features and to understand scene topographies simply from 5 partial observations. In comparison, GTM-SM generally fails to adapt the VAE prior and predict the belief of target sequences (refer to the supplementary material for further results).

Quantitative Evaluation. Adopting the same metrics as in Section 4.1, we compare the methods. As seen in Table 1.D-E, our method slightly underperforms in terms of localization in the Doom environment. This may be due to the approximate rendering process VizDoom uses for the depth observations, with discretized values not matching the game units. Unlike GTM-SM which relies on action encodings for localization, these unit discrepancies affect our observation-based method. As to the quality of retrieved and hallucinated images, our method shows superior performance ($c.f.$ additional saliency metrics in the supplementary material). While the current results are still far from visually pleasing, the proposed method is promising, with improvements expected from more powerful generative networks.

5 Conclusion

Given unlocalized agents only providing observations, our framework builds global representations consistent with the underlying scene properties. Applying prior domain knowledge to harmoniously complete sparse memory, our method can incrementally sample novel views over whole scenes, resulting in the first complete read and write spatial memory for visual imagery. We evaluated on synthetic and real 2D and 3D data, demonstrating the efficacy of the proposed method's memory map. Future work can involve densifying the memory structure and borrowing recent advances in generating high-quality images with GANs [42].

References

[1] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017.

[2] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, et al. Home: A household multimodal environment. *preprint arXiv:1711.11017*, 2017.

[3] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *arXiv preprint*, 2016.

[4] Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. Active neural localization. In *ICLR*, 2018.

[5] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[6] Siddharth Choudhary, Vadim Indelman, Henrik I Christensen, and Frank Dellaert. Information-based reduced landmark slam. In *ICRA*, 2015.

[7] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping. *IEEE robotics & automation magazine*, 13, 2006.

[8] SM Ali Eslami et al. Neural scene representation and rendering. *Science*, 360(6394), 2018.

[9] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016.

[10] Marco Fraccaro, Danilo Jimenez Rezende, Yori Zwols, Alexander Pritzel, et al. Generative temporal models with spatial memory for partially observed environments. *arXiv preprint:1804.09401*, 2018.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[13] Akshay Gore and Savita Gupta. Full reference image quality metrics for jpeg compressed images. *AEU-International Journal of Electronics and Communications*, 69(2):604–608, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[15] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM Trans. Graphics*, 35, 2016.

[16] Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *CVPR*, 2018.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[18] Dinghuang Ji, Junghyun Kwon, Max McFarland, and Silvio Savarese. Deep view morphing. In *CVPR*, 2017.

[19] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *ICCV*, 2009.

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

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[23] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[24] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *ICLR*, 2018.

[25] Emilio Parisotto, Devendra Singh Chaplot, Jian Zhang, and Ruslan Salakhutdinov. Global pose estimation with an attention-based recurrent network. *arXiv preprint*, 2018.

[26] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *CVPR*, 2017.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[28] Robert J Peters, Asha Iyer, Laurent Itti, and Christof Koch. Components of bottom-up gaze allocation in natural images. *Vision research*, 45(18):2397–2416, 2005.

[29] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech, Oriol Vinyals, et al. Neural episodic control. *arXiv preprint:1703.01988*, 2017.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[31] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[32] Dan Rosenbaum, Frederic Besse, Fabio Viola, Danilo J Rezende, and SM Eslami. Learning models for visual 3d localization with implicit mapping. *arXiv preprint*, 2018.

[33] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[34] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[35] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *ECCV*, 2018.

[36] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016.

[37] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *CVPR*, 2017.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[39] Zhou Wang and Qiang Li. Information content weighting for perceptual image quality assessment. *IEEE Trans. Image Processing*, 20(5):1185–1198, 2011.

[40] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *ACSSC*, 2003.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, 13(4):600–612, 2004.

[42] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[43] Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. Vizdoom competitions: Playing doom from pixels. *IEEE Transactions on Games*, 2018.

[44] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *arXiv preprint:1805.04409*, 2018.

[45] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. A comprehensive evaluation of full reference image quality assessment algorithms. In *ICIP*, pages 1477–1480. IEEE, 2012.

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

[47] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[48] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016.

Supplementary Material

In the following sections, we introduce further pipeline details for reproducibility. We also provide various additional qualitative results (Figures S3 to S6) and quantitative comparisons (Section B.3) on 2D and 3D datasets. A video is also attached, presenting our solution applied to the incremental registration of unlocalized observations and generation of novel views.

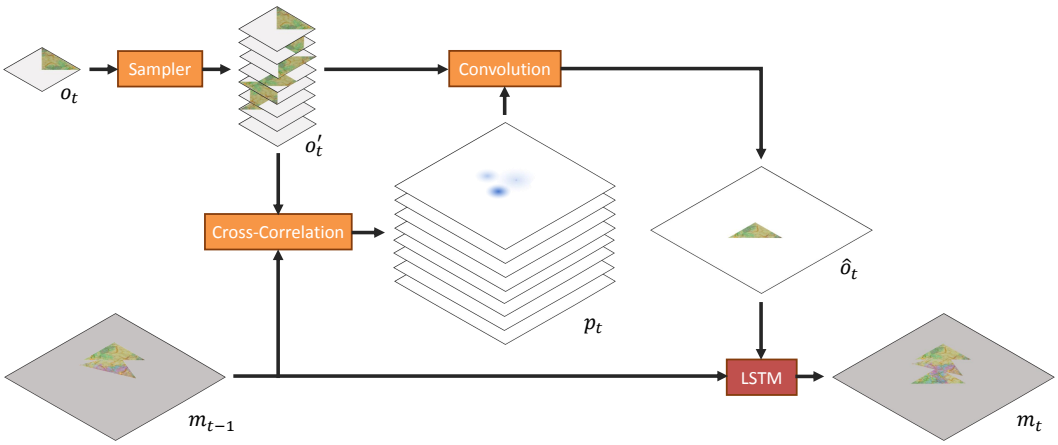

Figure S1: **Localization and memorization**, based on MapNet.

A Methodology and Implementation Details

This section contains further details regarding the several interlaced components of our pipeline and their implementation.

A.1 Localization and Memorization

A.1.1 Encoding Memories

Observations are encoded using a shallow ResNet [14] with 4 residual blocks. The encoder E is thus configured to output feature maps $x'_t \in \mathbb{R}^{n \times h' \times w'}$ with the same dimensions as the inputs $x_t \in \mathbb{R}^{c \times h \times w}$, *i.e.* $h = h', w = w'$.

As explained in Section 3.1, the projection of $x'_t \in \mathbb{R}^{n \times h' \times w'}$ (with features in the image coordinate system) into $o_t \in \mathbb{R}^{n \times s \times s}$, the representation of the agent's spatial neighborhood, is use-case dependent. For 2D image exploration, this operation is done by cropping x'_t into a square tensor $n \times s' \times s'$ with $s' = min(h',\ w')$, followed by scaling the features from $s' \times s'$ to $s \times s$ using bilinear interpolation.

For 3D use-cases with RGB-D observations, the input depth maps x^d_t are used to project x'_t into a 3D point cloud (after registering color and depth images together), before converting this sparse representation into a dense tensor using max-pooling. For the 3D projection, $\forall i \in \{0, ..., h-1\}$ and $\forall j \in \{0, ..., w-1\}$, each feature $x_{t,i,j} \in \mathbb{R}^n$ of x'_t receives the coordinates (x, y, z) similar to [16]:

$$z = x^d_{t,i,j} \quad ; \quad x = (j - c_x)\frac{z}{f_x} \quad ; \quad y = (i - c_y)\frac{z}{f_y} \tag{7}$$

with f_x, f_y the pixel focal lengths of the depth sensor, and c_x, c_y its pixel focal center (for KinectV2: $f_x = 366.193$px, $f_y = 365.456$px, $c_x = 256.684$px, $c_y = 207.085$px for 512×424 images).

Each set of coordinates is then discretized to obtain the neighborhood bin the feature belongs to. Given $s \times s$ bins of dimensions (x_s, z_s) in world units, the bin coordinates (x_b, z_b) of each feature are computed as follow:

$$x_b = \lfloor \frac{x}{x_s} \rfloor + \frac{s-1}{2} \quad ; \quad z_b = \lfloor \frac{z}{z_s} \rfloor + \frac{s-1}{2} \tag{8}$$

with $\lfloor \cdot \rfloor$ the integer flooring operation. Features projected out of the $s \times s$ area are ignored.

Finally, o_t is obtained by applying a max-pooling operation over each bin (*i.e.* keeping only the maximum values for features projected into the same bin). Empty bins result in a null value[1].

[1]Max-pooling over a sparse tensor (point cloud) as done here is a complex operation not yet covered by all deep learning frameworks at the time of this project. We thus implemented our own (for the PyTorch library [27]).

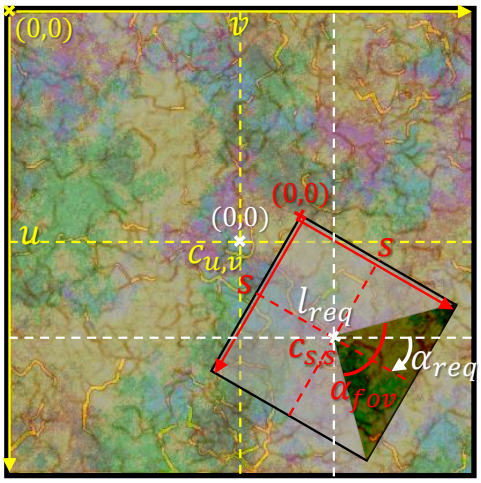

Figure S2: **Geometrical Memory Culling** (2D representation). The feature vector o_{req}, defining the observation for an agent positioned at l_{req} and rotated by an angle α_{req} in the allocentric system, is extracted from m_t through a series of geometrical transforms (rotation, translation, clipping, culling). Coordinates and distances in the allocentric coordinate system are represented in white; in yellow for the m_t matrix system; and red for the o_t one.

A.1.2 Localizing and Storing Memories

Once the features are localized and registered into the allocentric system (*c.f*. Figure S1), an LSTM is used to update the global memory accordingly, *c.f*. Section 3.1. Following the original MapNet solution [16], each spatial location is updated independently to preserve spatial invariance, sharing weights between each LSTM cell. The occupancy mask b_t of the global memory is updated in a similar manner, using an LSTM with shared-weights to update the memory mask with a binary version of o_t (*i.e*. 1 for bins containing projected features, 0 otherwise).

A.2 Anamnesis

A.2.1 Memory Culling

Figure S2 illustrates the geometrical process to extract features from the global map corresponding to the requested viewpoints and agent's view field angle, as described in Section 3.2. Note that for this step, one can use a larger view field than requested, in order to provide the feature decoder with more context (*e.g*., to properly recover visual elements at the limit of the agent's view field).

A.2.2 Memory Decoding

Similar to the encoder, we use a ResNet-4 architecture [14] for the decoder network, with the last convolutional layers parametrized to output the image tensors $x_{req} \in \mathbb{R}^{c \times h \times w}$ while the network receives inputs feature tensors $o_{req} \in \mathbb{R}^{n \times s \times s}$.

For the experiments where the global memory is directly sampled into an image , another ResNet-4 decoder is trained, directly receiving $m_t^h \in \mathbb{R}^{n \times u \times v}$ for input, returning $x_t^m \in \mathbb{R}^{c \times H \times W}$, and comparing the generated image with the original global image (L1 loss).

A.3 Mnemonic Hallucination

For simplicity and homogeneity[2], another network based on the ResNet-4 architecture [14] is used to fill the memory holes.

In order to improve the sampling of hallucinated features and the global awareness of this generator, we adopt several concepts from SAGAN [47]. The ResNet generator is therefore edited as follow:

[2]Our solution is orthogonal to the choice of encoding/decoding networks. More advanced architectures could be used.

- Spectral normalization [22] is applied to the weights of each convolution layer in the residual blocks (as SAGAN authors demonstrated it can prevent unusual gradients and stabilize training);
- To model relationships between distant regions, self-attention layers [47, 5, 23, 38] replace the two last convolutions of the network.

Given a feature map $o \in \mathbb{R}^{n \times u \times v}$, the result o_{sa} of the self-attention operation is:

$$o_{sa} = o + \gamma(W_h \star o)\sigma\big((W_f \star o)^\intercal (W_g \star o)\big)^\intercal \tag{9}$$

with $W_f \in \mathbb{R}^{\bar{n} \times n}$, $W_g \in \mathbb{R}^{\bar{n} \times n}$, $W_h \in \ltimes^{n \times n}$ learned weight matrices (we opt for $\bar{n} = {}^{n}/_{8}$ as in [47]); and γ a trainable scalar weight.

Following the generative adversarial network (GAN) strategy [12, 31, 33, 17], our conditioned generator is also trained against a discriminator evaluating the *realism* of feature patches o_t^h culled from m_t^h. This discriminator is itself trained against o_t (*real* samples) and o_t^h (*fake* ones). For this network, we also opt for the architecture suggested by Zhang *et al*. [47], *i.e*. a simple convolutional architecture with spectral normalization and self-attention layers.

Given this setup, the generative loss $\mathcal{L}_{hallu}$ is combined to $\mathcal{L}_{disc}$, a discriminative loss obtained by playing the generator H against its discriminator D. As a conditional GAN with recurrent elements, the objective this module has to maximize over a complete training sequence is therefore:

$$H^* = \arg\min_H \max_D \mathcal{L}_{disc} + \mathcal{L}_{hallu} + \mathcal{L}_{corrupt} \tag{10}$$

$$\text{with } \mathcal{L}_{disc} = \sum_{t=0}^{\tau} \big[\log D(x_t)\big] + \big[\log\Big(1 - D\big(x_t^h\big)\Big)\big] \tag{11}$$

A.4 Further Implementation Details

Our solution is implemented using the PyTorch framework [27].

Layer parameterization:

- Instance normalization is applied inside the ResNet networks;
- All Dropout layers have a dropout rate of 50%;
- All LeakyReLU layers have a leakiness of 0.2.
- Image values are normalized between -1 and 1.

Training parameters:

- Weights are initialized from a zero-centered Gaussian distribution, with a standard deviation of 0.02 ;
- The Adam optimizer [20] is used, with $\beta_1 = 0.5$;
- The base learning rate is initialized at $2e \times 10^{-4}$;
- Training sequence applied in this paper:
  1. Feature encoder and decoder networks are pre-trained together for 10,000 iterations;
  2. The complete memorization and anamnesis process (encoder, LSTM, decoder) is then trained for 10,000 more iterations;
  3. The hallucinatory GAN is then added and the complete solution is trained until convergence.

B Experiments and Results

Additional results are presented in this section. We also provide supplementary information regarding the various experiments we conducted, for reproducibility.

B.1 Protocol for Experiments

B.1.1 Comparative Setup

To the best of our knowledge, no other neural method covers agent localization, topographic memorization, scene understanding and relevant novel view synthesis in an end-to-end, integrated manner. The closest state-of-the-art solution to compare with is the recent GTM-SM project [10]. This method uses the differentiable neural dictionary (DND) proposed by Pritzel *et al*. [29] to store encoded observations with the predicted agent's positions for keys. To synthesize a novel view, the k-nearest entries (in terms of positions-keys) are retrieved to interpolate the image features, before passing it to a decoder network.

Unlike our method which localizes and registers together the views with no further context needed, GTM-SM requires an encoding of the agent's actions, leading to each new observation, as additional inputs. We thus adapt our data preparation pipeline for this method, so that the agent returns its actions (encoding the direction changes and step lengths) along the observations. At each time step, GTM-SM uses the provided action a_t to regress the agent's state s_t *i.e*. its relative pose in our experiments.

For a fair comparison with the ground-truth trajectories, we thus convert the relative pose sequences predicted by GTM-SM into world coordinates. For that, we apply a least-square optimization process to fit its predicted trajectories over the ground-truth ones *i.e*. computing the most favorable transform to apply before comparison (scaling, rotating and translating the trajectories). For our method, the allocentric coordinates are also converted in world units by scaling the values according to the bin dimensions (x_s, z_s) and applying an offset corresponding to the absolute initial pose of the agent.

B.1.2 Metrics for Quantitative Evaluations

As a reminder (*c.f*. main paper), the following metrics are applied, to evaluate the quality of the localization, the anamnesis, and the hallucination:

- The *average position error (APE)* computes the mean Euclidean distance between the predicted positions and their ground-truths for each sequence;
- The *absolute trajectory error (ATE)* is obtained by calculating the root-mean-squared error in the positions of each sequence, after transforming the predicted trajectory to best fit the ground-truth (giving an advantage to GTM-SM predictions through post-processing, as explained in Section B.1);
- The common *L1 distance* is computed as the per-pixel absolute difference between the predicted and expected values, averaged over each image (recalled and/or hallucinated);
- The *structural similarity (*SSIM*) index* [40, 41], prevalent in the assessment of perceptual quality [39, 45, 13], is computed over $N \times N$ windows extracted from the predicted and ground-truth images, as follow:

$$\mathrm{SSIM}(x, \bar{x}) = \frac{(2\mu_x\mu_{\bar{x}} + c_1) + (2\sigma_{x\bar{x}} + c_2)}{(\mu_x^2 + \mu_{\bar{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\bar{x}}^2 + c_2)} \tag{12}$$

  with x and $\bar{x}$ the windows extracted from the predicted and ground-truth images, μ_x and $\mu_{\bar{x}}$ the mean values of the respective windows, σ_x^2 and $\sigma_{\bar{x}}^2$ their respective variance, $c_1 = 0.001^2$ and $c_2 = 0.003^2$ two constants for numerical stability. The final index is computed by averaging the values obtained by sliding the windows over the whole images (no overlapping). We opt for $N = 5$ for the CelebA experiments, and $N = 13$ for the HoME-2D ones (*i.e*., splitting the observations in 9 windows). Note that the closer to 1 the computed index, the better the perceived image quality.

B.2 Navigation in 2D Images

B.2.1 Experimental Details

The aligned CelebA dataset [21] is split with $197,599$ real portrait images for training and $5,000$ for testing. Each image is center-cropped to 160×160px (in order to remove part of the background and focus on faces), before being scaled to 43×43px.

We build our synthetic HoME-2D dataset by rendering several thousand RGB floor plans of randomly instantiated rooms using the HoME framework [2] (room categories: "*bedroom*", "*living*", "*office*") and SUNCG data [34]. We use $8,960$ images for training and $2,240$ for testing, scaled to 83×83px.

For both experiments, sequences of observations are generated by randomly walking an agent over the 2D images. At each step, the agent can rotate maximum $\pm 90^\circ$ (for experiments with rotation) and cover a distance from $1/4$ to $3/4$ its view field radius. However, once a new direction is chosen, the agent has to take at least 3 steps before being able to rotate again (to favor exploration). The agent is also forced to rotate when one of the image borders is entering its view field.

Each training sequence contains 54 images for the CelebA experiments, and 41 for the HoME experiments. Both for GTM-SM [10] and our method, only 10 images are passed as observations to fill and train the topographic memory systems (using the provided ground-truth positions/orientations). The remaining 44 or 31 images (sampled by forcing the agent to follow a pre-determined trajectory covering the complete 2D environments) are used as ground-truth information for the hallucinatory modules of the two pipelines.

As described in Section 4.1, for each dataset we consider two different types of agents, *i.e*. with more or less realistic characteristics:

Simple agent. We first consider a non-rotating agent—only able to translate in the four directions—with a 360° view field covering an image patch centered on the agent's position. For CelebA experiments, this view field is 15×15px square patch; while for HoME-2D experiments, the view field reaches 20px away from the agent, and is therefore in the shape of a circular sector (pixels in the corresponding 41×41 patches further than 20px are set to the null value).

Advanced agent. A more realistic agent is also designed, able to rotate and to translate accordingly (*i.e*. in the gaze direction) at each step, observing the image patch in front of it (rotated accordingly). For CelebA experiments, the agent can rotate by $\pm 45^\circ$ or $\pm 90^\circ$ each step, and only observes the 8×15 patches in front (180° rectangular view field); while for HoME-2D experiments, it can rotate by $\pm 90^\circ$ each step, and has a 150° view field limited to 20px.

The first simple agent is defined to reproduce the 2D experiments showcasing GTM-SM [10]. While its authors present some qualitative evaluation with a rotating agent, we were not able to fully reproduce their results, despite the implementation changes we made to take into account the prior dynamics of the moving agent (*i.e*. extending the GTM-SM state space to 3 dimensions; the new third component of the state vectors s_{t-1} is storing the information to build a 2D rotation matrix, itself used with the translation elements to compute s_t). We adopt the more realistic agent to demonstrate the capability of our own solution, given its more complex range of actions and partial observations. Rotational errors with this agent are ignored for GTM-SM.

B.2.2 Additional Qualitative Results

As explained in the paper, our topographic memory module not only allows to directly build a global representation of the environments, but it also brings the possibility to use prior knowledge to extrapolate the scene content for the unexplored area. In contrast, GTM-SM stores each observation separately in its DND memory [29], and can only generate new views by interpolating between a subset of these entries with a VAE prior. This is illustrated in Figure 4 (comparing the methods on image retrieval from memory and on novel view synthesis) and Figure 5 (showcasing the ability of our pipeline to synthesize complete environments from partial views) in the main paper, as well as in similar Figures S3-S5.

B.3 Exploring Real 3D Scenes

B.3.1 Additional Quantitative Results

Additional evaluations were conducted on real 3D data, using the Active Vision Dataset (AVD) [1], in order to demonstrate the salient properties of the generated images (despite their lower visual quality).

First, the Wasserstein metric was computed between the Histogram of Oriented Gradients (HOG) descriptors extracted from the unseen ground-truth images and the corresponding predictions. GTM-SM [10] scored 1.1, whereas our method obtained 0.8 (the lower the better).

Second, we compared the saliency maps of ground-truth and predicted images [3], computing the area-under-the-curve metric (AUC) proposed by Judd *et al*. [19] and the Normalized Scanpath Saliency (NSS) [28]. GTM-SM [10] scored 0.40 for the AUC-Judd and 0.14 for the NSS, whereas our framework obtained 0.63 and 0.38 respectively (the higher the better for both metrics).

B.3.2 Additional Qualitative Results

For qualitative comparison with GTM-SM, we trained both methods on AVD dataset with the same setup. Challenges arise from the fact that the 3D environments are much more complex than their 2D counterparts, and more factors need to be considered in memorization and prediction.

Further qualitative results on the AVD test scenes are demonstrated in Figure S6. Given the same observation sequences (additional actions to GTM-SM) and requested poses, the predicted novel views are shown for comparison. Generally, GTM-SM fails to adapt the VAE prior and predict the belief of target sequences, while our method tends to successfully synthesize the room layout based on the learned scene prior and observed images.

Figure S3: **Qualitative comparison with GTM-SM** on CelebA and HoME-2D, in terms of pose / trajectory estimations and in terms of view generation (recovery of seen images from memory and novel view hallucination). Methods receive a sequence of 10 observations (along with the related actions for GTM-SM) from a non-rotating agent exploring the 83×83 2D image with a 360° circular view field of 20px radius. The methods then apply their knowledge to generate novel views.

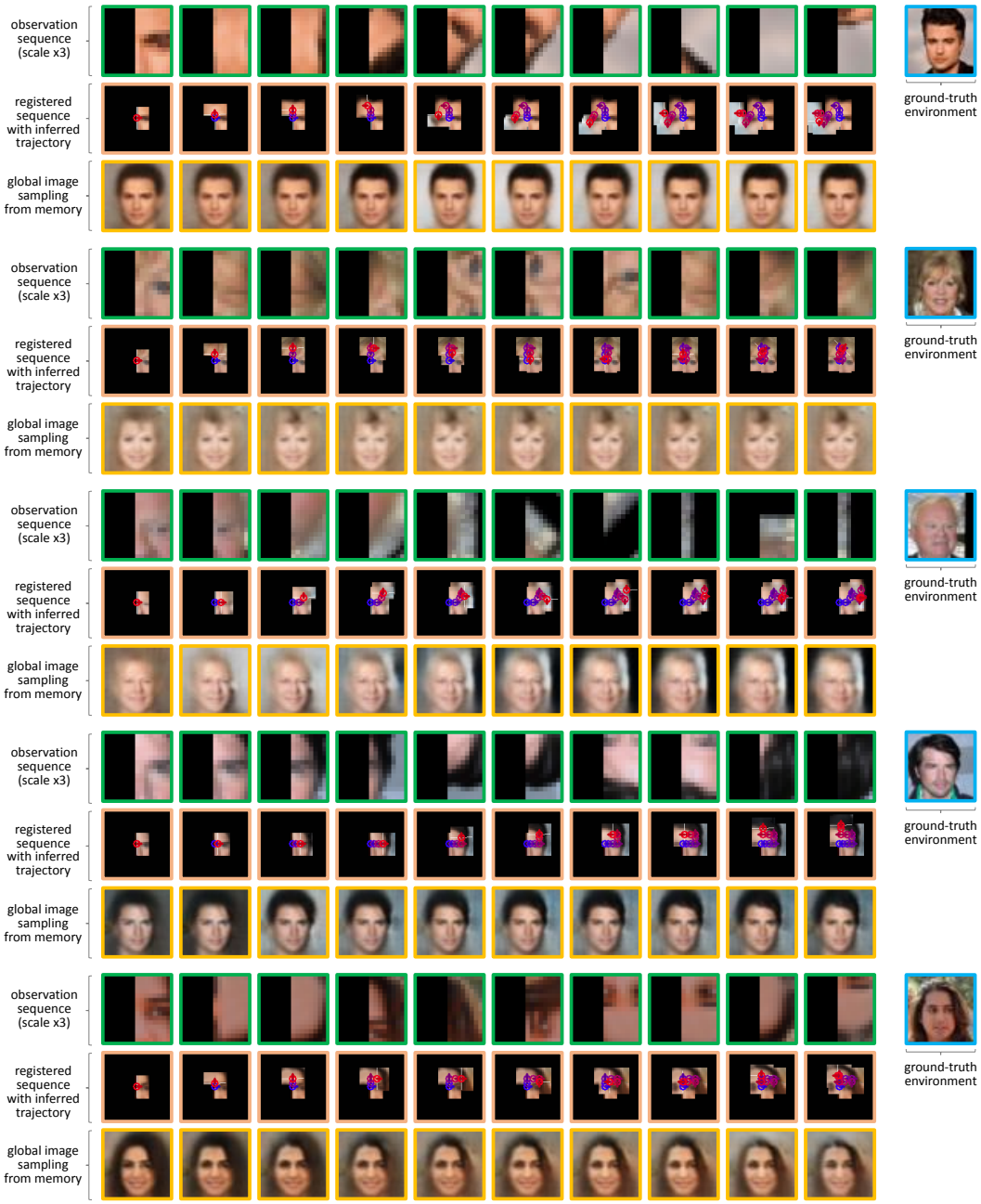

Figure S4: **Qualitative results on CelebA dataset**. with sequences of 10 observations from an agent able to rotate and translate every step, exploring the 43×43 2D image with a $180°$ view field of 8×15px (*i.e.* observing the image patch in front of it, rotated accordingly). After each step, the hallucinated features are adapted to blend with the new observations, until reaching convergence as the coverage increases.

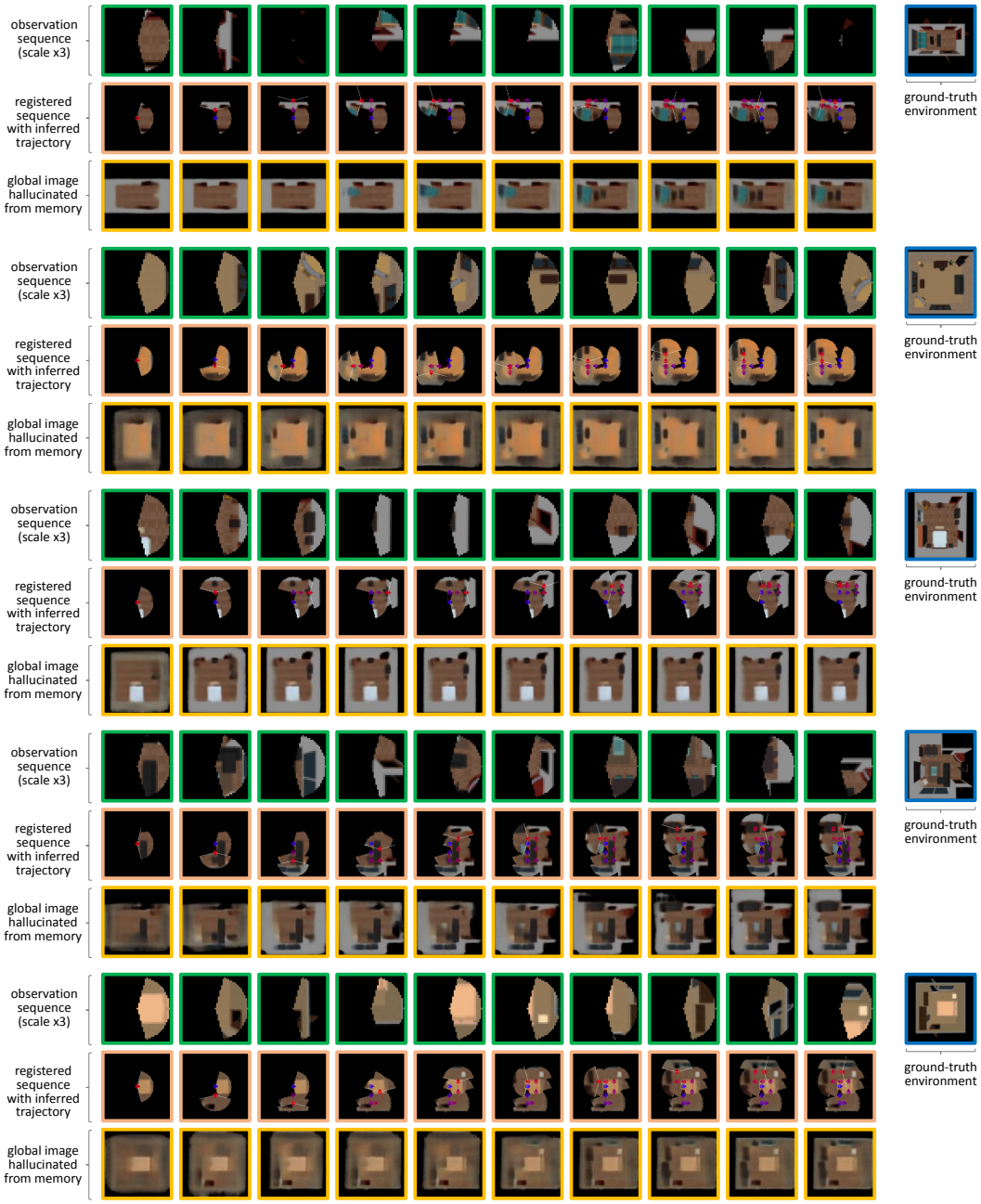

Figure S5: **Qualitative results on HoME-2D**. with sequences of 10 observations from an agent able to rotate and translate every step, exploring the 83×83 2D image with a $150°$ view field of 20px radius. After each step, the hallucinated features are adapted to blend with the new observations, until reaching convergence as the coverage increases.

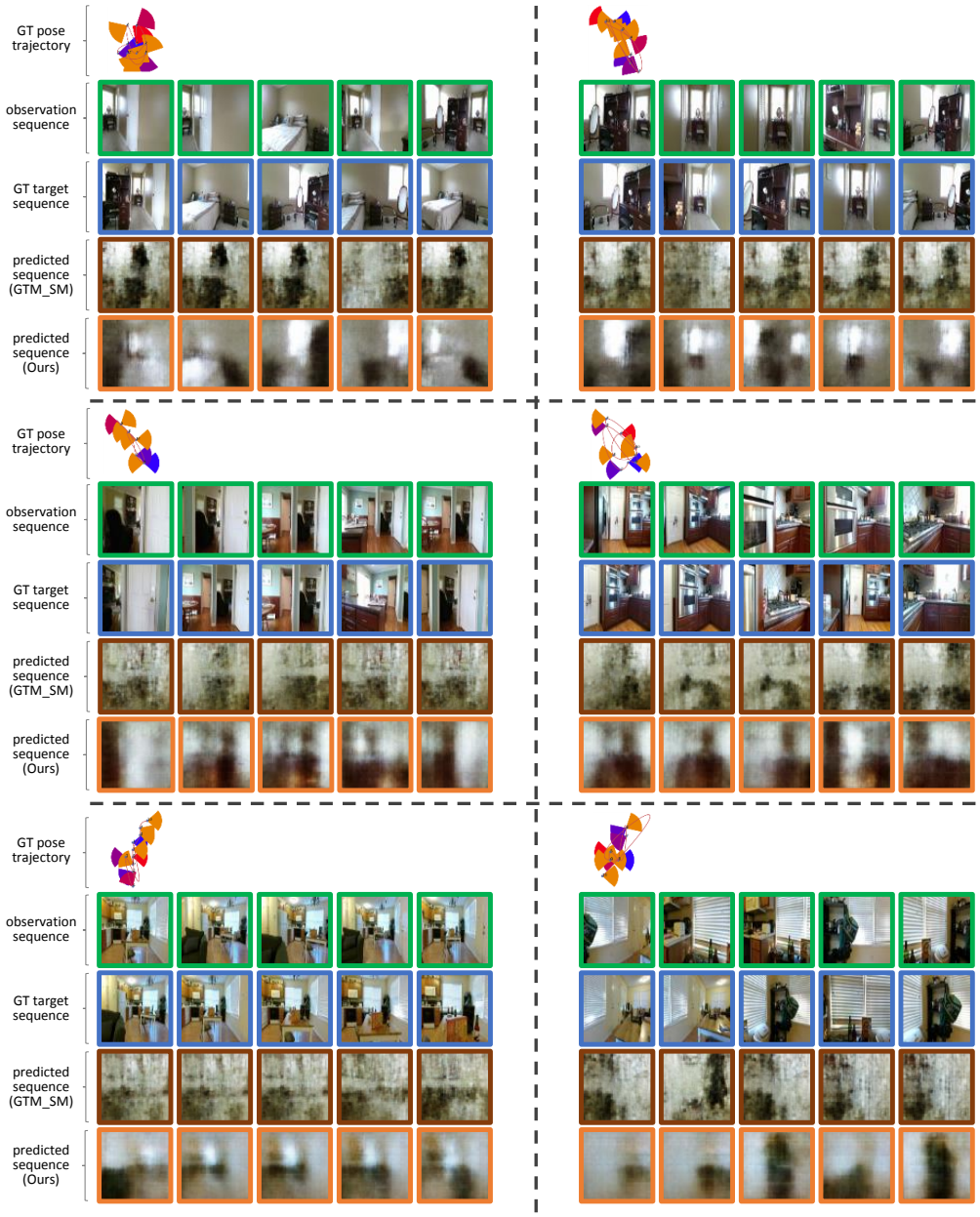

Figure S6: **Qualitative comparison with GTM-SM on AVD dataset**, in terms of pose / trajectory estimations and in terms of view generation (recovery of seen images from memory and novel view hallucination). Methods receive a sequence of 5 observations (along with the corresponding actions for GTM-SM) from an agent exploring the testing unseen scenes. The methods then apply their knowledge to generate novel views.