Expansive Latent Space Trees for Planning from Visual Inputs

Robert Gieselmann
KTH Royal Institute of Technology
Stockholm, Sweden
robgie@kth.se

Florian T. Pokorny
KTH Royal Institute of Technology
Stockholm, Sweden
fpokorny@kth.se

Abstract

Planning provides autonomous agents with the capability to solve long-horizon decision-making problems by evaluating predictions of consecutive future states. Visual observations are prevalent in many real-world applications due to their generic format and the availability of inexpensive sensors. We present a new method for long-horizon visual decision-making called Expansive Latent Space Trees (ELAST). Our method relies on self-supervised training via contrastive learning to obtain (a) a latent state representation and (b) a latent transition density model. ELAST generates paths in the learned space through sampling-based exploration within the estimated support region of the latent distribution. Unlike other methods, it circumvents the need for image generative models and does not rely on expert demonstrations. We embedded ELAST into a model-predictive control scheme and demonstrate its effectiveness for a set of simulated long-horizon visual control tasks.

1 Introduction

To perform challenging tasks in a real-world setting, autonomous agents must reason over many temporal steps often by processing high-dimensional sensor data. In the last decade, machine learning has significantly improved the state-of-the-art in vision-based robotics and perception [15]. Yet, most existing methods are limited to short-horizon problems and fail if the goal lies far ahead in the future. Reinforcement learning for instance becomes challenging in such settings due to the sparseness of rewards and the thereby complicated credit assignment [17]. Planning algorithms [13] on the other hand excel at solving temporally-extended decision-making problems. They predict a sequence of intermediate states towards a goal by initializing a search from a starting configuration. Common prerequisites for classical planning methods are the ability to measure distances between states and compact, i.e. low-dimensional, representations. The recent emergence of learning-based planning approaches has shown first success in lifting this concept to high-dimensional visual domains [19, 21, 24, 10, 18, 4, 8, 16, 7].

We present a new method for long-horizon latent planning based on recent advances in self-supervised representation learning and inspired by classical literature on sampling-based planning in robot configuration spaces [13, 9, 11]. Sampling-based approaches probe the search space in order to grow a tree or graph within the configuration space. The proposed planner builds upon this idea and generates paths through iterative exploration of a learned lower-dimensional state space. Instead of sampling new states from the underlying latent distribution, our method trains a dynamics model to expand the tree at its frontier circumventing costly training of generative models. For each new planning query, we initiate a new tree exploration starting from the current latent state. Our exploration strategy draws on Expansive Space Trees (ESTs) [9], hence we call our method Expansive

NeurIPS 2021Workshop: Self-Supervised Learning - Theory and Practice

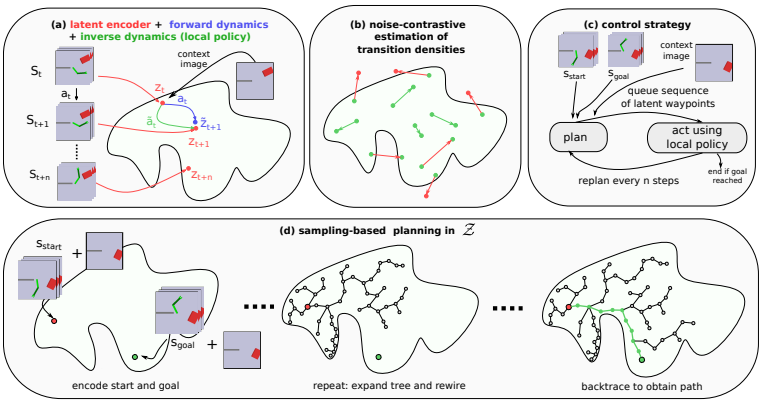


Figure 1: Conceptual overview of ELAST embedded into MPC control setting.

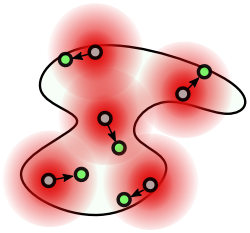


Figure 2: Illustration of the transition density estimation via NCE. Gaussian noise (red), starting state of transition (gray) and end point of the transitions (green).

Latent Space Trees (ELAST). Unlike EST however, we plan with an approximated dynamics model in a learned space for which the definition of distances is not trivial.

In this work, we explain how such a planner can be implemented using a self-supervised approach via contrastive learning of state representations and transition density estimators. We demonstrate in specific scenarios that visual long-horizon decision-making tasks may effectively be solved by sampling-based exploration in a learned lower-dimensional space. An initial set of experiments showed that ELAST yields significant performance improvements over existing baseline methods.

2 Expansive Latent Space Trees

Problem Definition This work studies long-horizon visual control tasks. We define a state space $\mathcal{S} = \mathbb{R}^{N \times C \times W \times H}$ for which N denotes the number of frames, C the number of color channels, W the width and H the height of single image frames. Instead of solving the control task directly in $\mathcal{S}$, we first map to a lower-dimensional latent space $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$. We use a goal space $\mathcal{G} = \mathcal{Z}$ and $d_\mathcal{A}$-dimensional continuous action space $\mathcal{A} = \mathbb{R}^{d_\mathcal{A}}$. We seek a goal-conditioned policy $\pi : \mathcal{Z} \times \mathcal{G} \to \mathcal{A}$ which navigates an agents towards a goal state. In addition to single environments, we also consider families of environments conditioned on a context vector $c \in \mathbb{R}^{d_c}$ capturing task-relevant information, e.g. the position of obstacles. Our aim is to solve tasks of the previous type given a collection of recorded interactions. Hence our method is designed for the offline settings in which we train self-supervised given fixed-size datasets of randomly generated trajectories.

Overview We propose ELAST which solves visual long-horizon control via sampling-based planning in a learned lower-dimensional space. A schematic overview is presented in Fig. 1. It consists of several independent modules including a state encoder ϕ, transition density model ψ, forward dynamics model h_f and policy π, each represented by neural networks. During execution time, ϕ maps the current and goal observations $s_{\text{start}}, s_{\text{goal}} \in \mathcal{S}$ to the latent encodings $z_{\text{start}}, z_{\text{goal}} \in \mathcal{Z}$ (Fig. 1a). For context-conditioned tasks, we additionally feed a context-vector c into the network [1]. The planning module (Fig. 1d) plans a sequence of intermediate latent states connecting z_{start} to z_{goal}. The quality of the paths is further improved using a custom tree-rewiring mechanism. Reconnecting nodes is enabled by the transition density model ψ (Fig. 1b) which we train using Noise-Contrastive Estimation [5, 6] in the latent space. Once a solution path is found, the controller module (Fig. 1c) queues the corresponding intermediate states to be achieved by a local policy π.

Contrastive and Predictive State Representation To overcome the challenges induced by the complexity of image observations, we instead plan in a lower-dimensional learned space. An encoder $\phi : \mathcal{S} \to \mathcal{Z}$ is defined which maps observations s_t into a latent space $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$ with $d_\mathcal{Z} \ll N \times C \times W \times H$. As described later, we desire an embedding which preserves the temporal distances between states. Since our data consists of random trajectories, we do not have access to the ground truth distances during training. Instead, we focus on pairwise similarity between states

[1] For brevity, the context c is omitted in the following.

and use Contrastive Predictive Coding (CPC) via InfoNCE [22] to learn an embedding which maps temporally neighboring states close together and uncorrelated ones far away from each other in $\mathcal{Z}$.

InfoNCE learns lower-dimensional representations of sequential input data and is theoretically motivated by preserving the mutual information between a context encoding and its future observations. Informally, it uses a classification objective which attempts to correctly identify a positive pair of samples, i.e. correlated ones, among negative ones, given a critic or similarity f. Similar to [20, 14, 23], we utilize a learned forward dynamics model $h_f(z_t, a_t)$ and consider positive pairs $(z_{t+1}, \hat{z}_{t+1})$. In this regard, $\hat{z}_{t+1}$ denotes the next state predicted given the current state z_t and action a_t. We compute the similarity between encodings based on the squared Euclidean distance and use a critic $f(s_t, s_{t+1}) = e^{-\|h_f(\phi(s_t), a_t) - \phi(s_{t+1})\|_2^2}$. Following [20], we encourage h_f to be consistent with the true latent dynamics by adding a MSE loss for h_f. The corresponding encoder loss is shown in Eq. 1

$$\mathcal{L}_{\phi,total} = -\mathbb{E}_{\mathcal{S}}\left[\log \frac{f(s_t, s_{t+1})}{\sum_{s_j \in \mathcal{S}} f(s_t, s_j)}\right] + \mathbb{E}_{\mathcal{Z}}\left[(z_{t+1} - h_f(z_t, a_t))^2\right] \tag{1}$$

The encoder obtained by optimizing Eq. 1 enforces locality of correlated states while shaping dynamics that are predictable by a model h_f. We argue that the latter encourages smoother embeddings which also facilitates the approximation of the local policy.

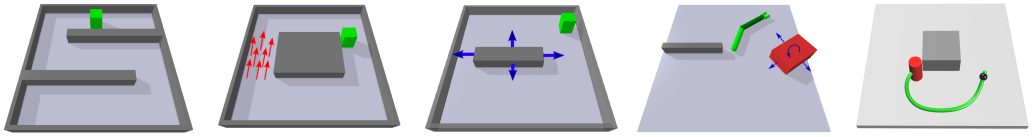
Causality of transitions via NCE Due to the symmetry of the critic f, our learned embedding cannot properly represent causal relationships between states. In other words, we organize subsequent states close together but lose the information in which direction transition are possible. Our planner however must be informed about the connectivity of states in order to rewire the tree and optimize the path. To recover this information, we estimate the density of latent transition $p(z_{t+1}|z_t)$ in a self-supervised fashion via Noise-Contrastive Estimation (NCE) [5, 6].

NCE trains an unnormalized density model and normalization constant by discriminating between samples from the original data distribution p_d and an auxiliary noise distribution p_n. For NCE to work properly, it is important that p_n lies within the support of p_d and is close to the unknown target distribution. Choosing a proper fit for p_n is generally difficult, we can however exploit that z_t and z_{t+1} lie close together in $\mathcal{Z}$. As shown in Fig. 2 we define p_n using multivariate Gaussians $\mathcal{N}(z_t, \Sigma)$ centered at z_t and set the value of Σ with respect to the average Euclidean distance between all z_t and z_{t+1} in the data (Fig. 2). We then use a model $\psi : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$, to estimate the log conditional density $\log p(z_{t+1}|z_t)$. After training ψ with NCE, we can detect invalid transitions by rejecting those whose log density lies below a certain threshold τ_t. In practice, we choose τ_t based on the split of the lower n-th percentile of densities of all latent transition in the data.

Dynamics Ensemble Due to the unboundedness of the search space $\mathbb{R}^{dz}$, erroneous latent transitions might result in endless exploration of undefined areas outside the latent distribution. After the encoder is trained, we replace the previous dynamics model with a more powerful ensemble estimator $H_f = \{h_f^1, .., h_f^k\}$ in order to reduce the number of detrimental transitions. Furthmore, we evaluate the ensemble's predictive uncertainty [12] to reject highly uncertain predictions during planning.

Tree Growth and Rewiring Given the encoder ϕ, forward dynamics ensemble H_f and transition density model ψ, we implement the tree expansion and rewiring strategy of ELAST. The planning starts by adding the initial encoding z_{start} to the tree. At each iteration step, we sample a state z_{expand} from the current tree. Given z_{expand} and a random action $a \sim \mathcal{A}$, we use H_f to expand the tree and generate a new state z_{new}. It is discarded if the associated transition density is low, i.e. $\psi(z_{\text{new}}, z_{\text{expand}}) < \tau_t$, or the predictive uncertainty $\text{Var}(\{h_f^i, i = 1..k\})$ exceeds a threshold τ_e. Otherwise, we continue and find the neighboring nodes of z_{new} within a radius of r_{neigh} that are reachable, i.e. pass the transition density test. For the set of resulting nodes, including z_{expand}, we pick the one that minimizes the overall cost to traverse from z_{start} to z_{new} and make it the new parent node of z_{new}. Lastly, we change z_{new} to be the parent node for each of its neighbor if its reduces its cost. The rewiring mechanism is illustrated in A.3

The above technique reconnects nodes to optimize the traveling distances to all nodes in the tree. We further encourage quick exploration by sampling z_{expand} weighted by the inverse number of reachable neighbors of each node. Once the tree reaches the vicinity of z_{goal}, we backtrace to determine the



(a) BlockS (b) BlockAsym (c) BlockParam (d) PlanarRobot (e) CableEnv

Figure 3: Illustration of the evaluation environments. The red arrows in (b) display asymmetric dynamics. The blue arrows in (c) and (d) indicate obstacle variations captured by the task context.

(a) BlockS (b) BlockAsym (c) BlockParam (d) PlanarRobot (e) CableEnv

Figure 4: Latent paths planned with ELAST visualized in the Isomap embedding

path. Our optimization strategy presents an adapation of the one in [11]. It has been used to design asymptotically-optimal planners for sampling-based path-planning in robot configuration spaces.

3 Experiments

Setup We embed ELAST in a Model Predictive Control (MPC) scheme and evaluate its performance for the simulated visual control tasks in Fig. 3. In the *BlockS* environment, a block agent navigates through a long S-shaped corridor. In *BlockAsym*, the agent must take into account the unidirectional stream on the left side of the workspace. This task is particularly interesting due to underlying asymmetric dynamics. In *BlockParam* and *PlanarRobot*, a block agent respectively robot arm is navigated around an obstacle. The configuration of the obstacle may vary hence representing context-conditioned tasks. Further information about the evaluation environments is provided in A.1.

Table 1: Performances in terms of success rates (average over three independent runs)

Method	BlockS		BlockAsym	BlockParam	PlanarRobot	CableEnv
	medium	hard				
ELAST (ours)	0.97 ± 0.01	0.91 ± 0.04	0.99 ± 0.0	0.81 ± 0.03	0.76 ± 0.0	0.93 ± 0.01
CPC-CEM	0.39 ± 0.03	0.03 ± 0.02	0.21 ± 0.09	0.13 ± 0.01	0.66 ± 0.0	0.03 ± 0.01
CPC-BC	0.18 ± 0.02	0.0 ± 0.0	0.0 ± 0.0	0.42 ± 0.03	0.03 ± 0.02	0.03 ± 0.0
HTM	0.07 ± 0.03	0.0 ± 0.0	0.0 ± 0.0	0.16 ± 0.02	0.19 ± 0.03	0.02 ± 0.0
Visual-BC	0.25 ± 0.08	0.0 ± 0.0	0.04 ± 0.06	0.35 ± 0.02	0.56 ± 0.01	0.02 ± 0.01
Random	0.07 ± 0.02	0.0 ± 0.0	0.0 ± 0.0	0.03 ± 0.02	0.04 ± 0.01	0.02 ± 0.01

Results As shown in Table 1, ELAST clearly outperforms the baselines in terms of average success rates. One reason is that it explores the latent space more efficiently which is crucial in long-horizon settings. The Cross-Entropy-Method (CEM) [2] fails to reach far distant goals since it relies on trajectory shooting without further encouraging exploration. Fig. 4 presents several examples of latent paths planned by ELAST, projected into the 2D Isomap embedding.

4 Conclusion

We presented an application of self-supervised learning for planning given high-dimensional image observations. We took inspiration from robot motion planning to design a planner that grows a search tree within the estimated support of the latent distribution. A key insight is that density estimation via NCE can be used to predict the temporal direction of transitions in the latent space. We embedded ELAST in a model-predictive control setting and demonstrated its effectiveness for challenging long-horizon tasks in simulation.

Acknowledgments

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Agx dynamics - real-time multi-body simulation. `http://www.algoryx.se/agx-dynamics/`, 2021.

[2] Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. Chapter 3 - the cross-entropy method for optimization. In C.R. Rao and Venu Govindaraju, editors, *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pages 35–59. Elsevier, 2013.

[3] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[4] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[5] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[6] Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012.

[7] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[8] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 09–15 Jun 2019.

[9] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3, pages 2719–2726 vol.3, 1997.

[10] B. Ichter and M. Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, 2019.

[11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[14] Nir Levine, Yinlam Chow, Rui Shu, Ang Li, Mohammad Ghavamzadeh, and Hung Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. In *International Conference on Learning Representations*, 2020.

[15] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[16] Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for zero-shot visual planning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6259–6270. PMLR, 13–18 Jul 2020.

[17] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[18] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[19] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

[20] Rui Shu, Tung Nguyen, Yinlam Chow, Tuan Pham, Khoat Than, Mohammad Ghavamzadeh, Stefano Ermon, and Hung Bui. Predictive coding for locally-linear control. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8862–8871. PMLR, 13–18 Jul 2020.

[21] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4732–4741. PMLR, 10–15 Jul 2018.

[22] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[23] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *CoRR*, abs/2003.05436, 2020.

[24] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

A Appendix

A.1 Environment Implementation and Testing Parameters

All environments, except *CableEnv* were implemented in PyBullet [3]. The *CableEnv* was created using the AGX dynamics physics engine [1].

Problems without context-conditioning For environments without a context-conditioning, training datasets consist each of 1000 random trajectories of length 25. Actions for data collection were generated by initially sampling an action uniformly and then adding Gaussian noise during the subsequent steps.

The test data contains start and goal configurations that were generated far distant from each other to ensure difficult long-hotizon settings. We evaluate on 100 unseen test cases. For *BlockS* we sample start and goal in a way that they are separated by one wall (medium) or two walls (hard). For *BlockAsym* we sample start and goal on different sides of the corridor and such that the agent cannot use the stream to reach the goal. The testing start and goal states in *CableEnv* are generated on opposite sides of the obstacle.

Context-conditioned Tasks The generated data in context-conditioned environments contains 100 random contexts in both tasks. For each context, we generate 25 trajectories of length 20. To generate vector representations of the context, we use the latent vectors generated by training a convolutional autoencoder given raw image observation of the scene.

The test set consists of newly sampled contexts. In the *BlockParam* domain we sample test start and goal states on opposite sides of the obstacle. The test data in *PlanarRobot* is generated by ensuring a minimal angular distance of $\pi/2$ between the rotational angle of the base joints.

Table 2: Maximum number of steps for testing

Environment	Max Steps
BlockS (medium)	100
BlockS (hard)	100
BlockAsym	100
BlockParam	50
PlanarRobot	75
CableEnv	50

A.2 Expansive Latent Space Trees - Further Details and Implementation

A.3 Tree Rewiring Mechanism

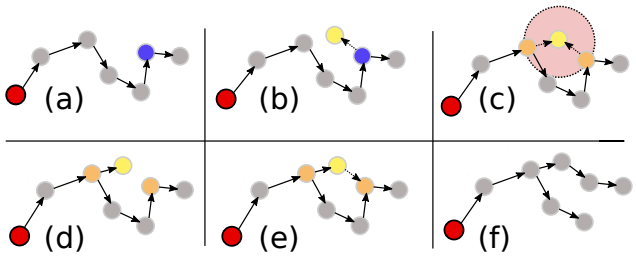


Figure 5: Tree growth and rewiring mechanism. Repeat: **(a)** select random node z_{expand} (blue) in the exisitng tree **(b)** create new node z_{new} (yellow) from z_{expand} using random action and learned forward model h_f (or ensemble H_f) **(c)** find set of nodes $Z_{neigh} = \{z_0, z_1, ..\}$ (yellow) in local neighborhood around z_{new} **(d)** change parent of z_{new} if there exist a node in Z_{neigh} with lower total travelling cost and that transitions to z_{new} **(e),(f)** alter parent of nodes in Z_{neigh} if transitioning from z_{new} is possible and reduces travelling cost.

A.3.1 Network Architecture and Training

We used the same network architectures in all experiments. The encoder parameters are show in Table 3. In *BlockS* and *BlockAsym*, we use latent spaces of dimension 8 and 16 for the remaining environments. The density model, forward model and policy use four hidden layers each consisting of 64 neurons and LeakyRelu activation function.

Table 3: Hyperparameters of encoder ϕ

Parameter	Value
Filter	[16,16,16,32,64,64]
Kernels	[3,4,4,4,4,4]
Strides	[1,2,1,2,2,2]
Activation	LeakyRelu
Dense Layers	[256,128]
Latent Dimension	8 or 16

A.3.2 Planning Module

The hyperparameters of our planning module are shown in Table 4. We use $n_{\text{iter}} = 2500$ sampling iterations in *BlockParam*, *PlanarRobot* and *CableEnv* and $n_{\text{iter}} = 5000$ otherwise. During exploration, we sample nodes uniformly with probability $p_{\text{uniform}} = 0.2$. With probability p_{bias}, nodes are sampled weighted by the inverse of their number of neighbors and with p_{goal} we pick the node that lies closest to the goal (Euclidean distance). To sparsen the tree, we discard new states that are closer than r_{discard} to any node in the current tree.

Table 4: Hyperparameters of planning module

Parameter	Value	Description
n_{iter}	2500 or 5000	Num. of samples states (includes rejected ones)
p_{uniform}	0.2	Probability of sampling node uniformly
p_{bias}	0.78	Probability of using biased sampling
p_{goal}	0.02	Probability of using sampling node closest to goal
r_{neigh}	avg. neigh. dist in $\mathcal{Z}$	Radius to find neighbors for rewiring
r_{discard}	0.25 x avg. neigh. dist in $\mathcal{Z}$	Radius to discard states
τ_{t}	2-th percentile	Threshold for density test. Reject transition if below
τ_{e}	98-th percentile	Threshold for uncertainty test. Reject state if above.

A.4 Implementation of Baselines

A.4.1 CPC-CEM

We apply CEM within the learned CPC embedding using the parameters below.

Table 5: Parameters of CEM Planner

Parameter	Value
Planning horizon	25
Samples per iteration	100
Iterations	5
Elite size	10

A.4.2 Hallucinative Topological Memory

We used the implementatio of HTM from `https://github.com/thanard/hallucinative-topological-memory`. Similar to [16] we use a latent space of size 100

and 500 samples to generate the map of images. During the experiments we noticed that the predicted image paths often contained shortcut connections which rendered the entire trajectory infeasible for the policy.

A.4.3 Behavioral Cloning

For latent space behavioral cloning we used a neural network policy consisting of four layer with 64 neurons each and LeakyRelu activation functions.