Latent Planning via Expansive Tree Search

Anonymous Author(s)
Affiliation
Address
email

Abstract

Planning provides autonomous agents the capability to solve long-horizon decision-making problems by evaluating predictions of the future. Yet, many real-world settings exhibit high-dimensional state spaces along with unknown transition dynamics, rendering traditional planning approaches infeasible. The idea behind latent planning is to overcome those challenges by instead solving the decision-making task in a lower-dimensional embedding space. Most existing latent planners employ shooting or collocation-based trajectory optimization techniques which are known to fail in very long-horizon and highly non-convex settings. In this work, we formulate latent planning as search to discover paths between far distant states in high-dimensional and long-horizon goal-reaching scenarios. Inspired by classical sampling-based motion planning algorithms, we designed a method which explores into the latent space deeper by iteratively growing and optimizing a search tree directed towards undiscovered areas while being constrained to the estimated data support region. Our method, called Expansive Latent Space Trees (ELAST), relies on self-supervised training via contrastive learning to obtain (a) a latent state representation and (b) a latent transition density model. We embed ELAST into a model-predictive control scheme and demonstrate significant performance improvements compared to existing baselines given challenging visual control tasks in simulation, including the navigation for a deformable object.

1 Introduction

To perform challenging control tasks in real-world environments, intelligent agents must reason across many temporal steps, often relying on high-dimensional sensor data such as images. Over the past decade, machine learning has significantly improved the state-of-the-art in image-based robotics and perception [31, 8, 2, 33, 16, 14]. However, existing methods are often limited to problems with relatively short time horizons and fail when the target is too far in the future. Reinforcement learning (RL), for example, is vulnerable to such scenarios due to sparse reward feedback and the resulting complexity of credit assignment [34, 15]. Planning algorithms [27], on the other hand, excel at solving temporally-extended decision problems, but typically require compact representations, state distance metrics and perfect knowledge of transition dynamics.

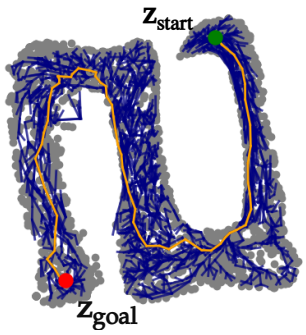


Figure 1: Sampling-based latent exploration with ELAST. A tree (blue) is interatively grown and optimized while being bound to the estimated latent support region (gray).

Recent work proposes a mixture of classical planning and deep learning methods to enable planning from complex high-dimensional observations. Current approaches can be broadly divided into planning (a) directly in the observation space [41, 7, 32] and (b) in learned latent spaces [13] [40]. The first variant

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

usually relies on generative modeling of video sequences to synthesize feasible image paths. The inherently high computational cost of training and deploying high-dimensional generative models, as well as the burden to make visually accurate predictions over long time horizons, present key limitation of this type of method. Latent planning, on the other hand, maps the control problem to a low-dimensional embedding space to facilitate the approximation of the dynamics. The idea of solving the planning task entirely in a lower-dimensional space is appealing because it bypasses the computational bottleneck of synthesizing observations and therefore has the potential to rapidly compute paths even in long-horizon settings.

Most existing latent planners are based on trajectory optimization tools originally developed in the optimal control literature [22]. In particular, gradient-free shooting-based strategies such as the Cross-Entropy Method (CEM) [3] or Model-Predictive Path Integral control (MPPI) [52] attracted much attention due to various applications in model-based reinforcement learning [13] [36]. Also, numerical trajectory optimization through direct collocation [22] has been used by [40] in the context of planning over latent spaces. Despite their recent popularity, the aforementioned techniques are known to suffer from local minima when the optimization landscape is highly non-convex [46] and therefore often fail to compute feasible paths for distant goals in geometrically complex latent spaces.

In this work, we pursue the intriguing concept of planning in latent spaces and focus on improving existing techniques by formulating latent planning as search within the continuous latent state space. We present **E**xpansive **LA**tent **S**pace **T**rees (ELAST) which solves high-dimensional goal-reaching tasks through an explorative tree search biased towards undiscovered regions of the latent manifold. Our method is inspired by single-query sampling-based planners [27] in the field of robot motion planning, in particular the asymptotically optimal versions of Rapidly-exploring Random Trees (RRT) [28] and Expansive Space Trees (EST) [17]. Motion planners are typically used for navigating robots through geometrically complex environments, analogously our motivation is to introduce similar concepts in latent spaces. However, it should be noted that direct application of these methods to our setting is **not possible because state metrics, sampling distribution, collision checks and other important quantities are difficult to define in learned spaces**. To overcome this challenge, we develop a method that uses self-supervised contrastive learning to first train an embedding that locally preserves certain geometric properties that favor planning, and in a second step approximate dynamics and state connectivity from random interactions with an environment. One of our key findings is that, given the contrastive embedding, we can approximate the conditional transition densities by Noise-Contrastive Estimation (NCE) [10, 11] in the latent space.

The main contribution of this work is the presentation of ELAST, a new latent planning algorithm designed for long-horizon goal-reaching. Compared to exisiting approaches that plan in the observation space, our method is lightweight in design and does not require costly training of high-dimensional generative models. To the best of our knowledge, we are the first to present latent planning using explorative tree search without the need of expert data or the underlying ground truth state space. Unlike static map-based approaches [41] [7] [9], we show that our planner is compatible with parameterized environments such as conditioning on the position of obstacles. We demonstrate the effectiveness of ELAST by integrating it into a model-predictive control scheme and comparing it to existing latent planning bases in challenging visual planning tasks in simulation. Our experimental evaluation shows a significant performance improvement in terms of achieved downstream task success rates and the quality of the resulting path.

2 Related Work

Planning in Images Spaces A large body of prior works learn video dynamics to solve visual planning tasks [8, 6, 54, 37, 48]. Visual Foresight [6], for example, generates predictions of future frames to optimize next-step actions through shooting-based optimization. [48] extends this idea with offline Q-learning to obtain improved cost-to-go metrics in long-horizon settings. [37] uses a hierarchical architecture to produce subgoal images for temporally-extended navigation tasks. The approaches in [7, 18, 26] learn a distance metric using value-based reinforcement learning to construct a graphical representaion of the observation space. Paths are then generated by solving the shortest path problem on the graph. In a similar fashion, [41] uses self-supervised training of a connectivity classifier to construct a map of neighboring image states. [32] builds upon this idea but uses an energy-based model to approximate the traversability between states. The methods presented in [24, 50] use a map-free approach which plans visual paths by predicting intermediate frames

using a generative adversarial network. In this work, we study a fundamentally different approach to long-horizon visual decision-making by instead planning in latent spaces which effectively avoids expensive training and deployment of video generative models.

Latent Planning and Control The idea behind latent planning and control is to provide manageable and lower-dimensional representations to simplify decision-making and dynamics learning. [51] train a linear dynamics model to achieve locally-linear control from images in a learned embedding. [30, 43, 39] investigate contrastive-predictive coding to learn useful representations for control. [13] uses a reconstruction-based visual embedding and plans actions using shooting-based trajectory optimization via the Cross-Entropy Method (CEM) [3]. Similary, [36] uses Model-Predictive Path Integral Control (MPPI) [52] with deep dynamics models for dexterous robot manipulation tasks. [12] use contrastive representations and implement a model-based RL agent for latent planning from images. [40] recently introduced collocation-based trajectory optimization [22] to model-based reinforcement learning in latent spaces. The methods in [45, 56] approach visual control tasks via gradient-based planning in a learned embedding space. [38] plans sequences of coarse subgoals by solving the shortest-path problem through direct optimization on the latent manifold. [19] implement a sampling-based motion planner within a learned lower-dimensional space for robot motion planning from image inputs, yet relying on the existence of a state sampler and collision checking mechanism. [4] follows the concept of sampling-based motion planning [27] in latent spaces yet over sub-tasks which requires access to demonstration data and a task sampler.

Our method employs sampling-based planning in a learned space similar to the works of [19] [4]. In contrast, we do not require expert data, collision checking oracles or any kind of access to the underlying ground truth state space or sub-tasks. ELAST generates a search tree for every new planning query. Unlike map or graph-based approaches [7, 18, 26, 41], this provides more flexibility and enables planning beyond static environments, e.g. parameterization on obstacle configurations.

3 Preliminaries

Sampling-based Motion Planning In the motion planning literature, the robot configuration space is defined as the space of all possible configurations, i.e., the specifications that accurately describe the state of a robot system [27]. Sampling-based motion planners solve continuous robot motion planning tasks by performing a search that probes the configuration space through sampling [27]. A prominent representative of sampling-based algorithms are Rapidly-exploring Random Trees (RRT) [28]. It grows a search tree by iteratively adding new samples drawn from the configuration space. At each step, a new state is connected to its nearest neighbor in the tree if the corresponding transition is valid. Instead of sampling directly from the state distribution, Expansive Space Trees (EST) [17] builds a search tree by randomly expanding nodes located in sparsely represented regions of the configuration space. Improved versions of EST and RRT have been presented in [21] which achieve asymptotically-optimal paths as the number of iterations increases. Classical motion planners operate on well-described metric spaces and assume access to collision detection and state samplers. In this work, we bridge the gap between sampling-based robot motion planning and latent planning.

Noise-Contrastive Estimation Density estimation is a ubiquitous task in machine learning and statistics. Unnormalized density models, often known as energy-based models, offer flexible parameterizations for instance using expressive neural networks architectures. A recent introduction to energy-based model optimization is given in [44]. In this regard, Noise-contrastive estimation (NCE) [10, 11] has become a widely-used training method due to its balance between statistical efficiency and computation. NCE trains a density model by distinguishing between points x from the original data distribution p_d and samples drawn from an auxiliary noise distribution p_n. The corresponding maximum likelihood objective, shown in Eq. 2, resembles nonlinear logistic regression. NCE simultaneously estimates an unnormalized model p_m^0 and normalization constant b, which allows to retrieve normalized density estimates p_m , where $\ln p_m(u) = p_m^0(u) + b$. Note that $\mathbb{E}$ is in practice computed as an empirical mean from the training data.

$$m(u) = \frac{1}{1 + \nu \exp(-G(u))} \quad \text{with} \quad G(u) = \ln p_m(u) - \ln p_n(u) \tag{1}$$

$$\mathcal{J}_{\text{NCE}} = \mathop{\mathbb{E}}_{x \sim p_d} \big[\ln[m(x)] \big] + \nu \mathop{\mathbb{E}}_{y \sim p_y} \big[\ln[1 - m(y)] \big] \tag{2}$$

To ensure good approximation results, it is crucial that the support of p_n contains p_d without deviating too much from it. Moreover, p_n should be computationally efficient to evaluate for any input. A central problem in NCE is therefore to choose a proper noise for the unknown data distributions. For detailed derivations, we refer to [10, 11].

Contrastive Predictive Representations Contrastive representation learning compares among data samples to learn embeddings which map similar data samples close together and dissimilar ones far apart [29]. Contrastive Predictive Coding (CPC) [49] uses this principle to learn latent representations $\mathcal{Z}$ for sequential data $\mathcal{X}$. The InfoNCE objective (Eq. 3) was derived which attempts to maximally preserving the mutual information between the k-step future observations x_{t+k} and context representations y_t which pool prior information $z_{\leq t}$ in the latent space $\mathcal{Z}$. Inspired by NCE, it identifies correlated pairs drawn from $p(x_{t+k}|y_t)$ among uncorrelated ones, given a user-defined similarity $f(x_{t+k}, y_t)$. [49] use $f(x_{t+k}, y_t) = \exp(z_{t+k}^T W_k y_t)$ which computes similarity by taking the dot product between the latent encodings z_{t+k} and their corresponding future linear predictions $W_k^T y_t$.

$$\mathcal{L}_{\text{InfoNCE}} = -\mathop{\mathbb{E}}_{\mathcal{X}} \left[\log \frac{f(x_{t+k}, y_t)}{\sum_j f(x_j, y_t)} \right] \tag{3}$$

CPC has been shown to learn smooth representations in various control settings [43, 30, 53, 32]. We will utilize it to recover embeddings that are compatible with our downstream planning algorithm.

4 Expansive Latent Space Trees

We study visual control tasks given Markovian image sequence observations $\mathcal{S}{=}\mathbb{R}^{N\times C\times W\times H}$, where N denotes the number of frames, C the number of channels, W the width and H the height of single frames. Instead of directly working in image space, we solve the control task in a lower-dimensional latent space $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$. In the context of this work, we are interested in long-horizon goal-reaching tasks and define a goal space $\mathcal{G} = \mathcal{Z}$ and $d_\mathcal{A}$-dimensional continuous action space $\mathcal{A} = \mathbb{R}^{d_\mathcal{A}}$. The unknown goal-conditioned policy $\pi : \mathcal{Z} \times \mathcal{G} \to \mathcal{A}$ locally navigates towards a goal. In addition to single environments, we also consider families of environments conditioned on raw context observations $c \in \mathcal{C}$ (e.g. images) capturing task-relevant information, e.g. the position of obstacles. We seek a method that is trained from randomly collected trajectory data without access to optimal behavior, underlying dynamics or distances.

In this section, we present ELAST which solves the described type of control problems through explorative tree search within $\mathcal{Z}$. A schematic overview is presented Fig. 2. Our method consists of a state encoder ϕ, transition density model ψ, forward dynamics model h_f and local policy π, each represented using 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. 2a). For context-conditioned tasks, we additionally feed a context-vector c into the network. The planning module (Fig. 2d) plans a sequence of intermediate latent states connecting z_{start} to z_{goal}. To guarantee high quality of solution paths in terms of temporal path length, the search tree must be optimized during planning. For that, we introduce a custom rewiring strategy (Sec. 4.3) which relies on state connectivity estimation enabled by our transition density model ψ (Fig. 2b). The planner returns a discretized n-step piece-wise linear solution path in $\mathcal{Z}$ to the controller module which then queues the corresponding intermediate states to be achieved by π.

4.1 Geometric Properties of the Encoder

We define an encoder $\phi : \mathcal{S} \times \mathcal{C} \to \mathcal{Z}$ which maps observations s_t conditioned on a context c into a latent space $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$ with $d_\mathcal{Z} \ll N{\times}C{\times}W{\times}H$. For brevity, we for now focus on environments without a task-specific context, hence $\phi : \mathcal{S} \to \mathcal{Z}$. As further described in Sec. 4.3, our planner performs nearest-neighbor lookups during tree rewiring and node sampling. To allow for simple and efficient neighborhood computations, we desire ϕ to preserve the proximity of temporally subsequent states in $\mathcal{Z}$. Yet, defining a global distance metric to train such an encoder is non-trivial since it needs to account for the semantic information contained in the images. Moreover, it cannot be easily inferred from our data which consist of sub-optimal trajectory data. Instead, we rely on the knowledge of continuity between observations to enforce proximity of temporally adjacent states. More specifically, we employ contrastive learning via CPC (Sec. 3) and consider positive pairs

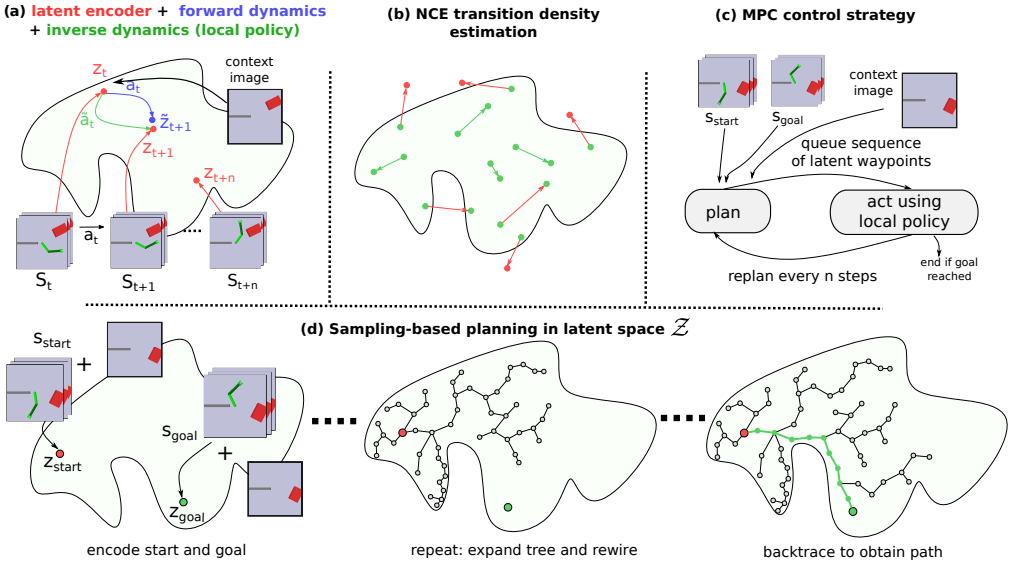


Figure 2: Overview of the different modules of ELAST. (a) training of latent encoder, dynamics and policy, (b) noise-contrastive estimation of transition densities $p(z_{t+1}|z_t, c)$, (c) model-predictive control framework, (d) planning via explorative tree search in $\mathcal{Z}$.

$(z_{t+k}, \widetilde{z}_{t+k})$ where $\widetilde{z}_{t+k}$ denotes the predicted future latent state of z_t given a sequence of actions $a_{t:t+k-1}$. To obtain $\widetilde{z}_{t+k}$ we simultaneously train an auto-regressive dynamics model $h_f : \mathcal{Z}\times\mathcal{A}\rightarrow\mathcal{Z}$ supervised given the transitions in the data. A similarity based on the squared Euclidean distance is used, hence $f(s_{t+k}, s_t) = e^{-\|\widetilde{z}_{t+k} - z_{t+k}\|_2^2}$. The encoder loss $\mathcal{L}_\phi$ is depicted in Eq. 4. In accordance with the observations in [43, 30, 53] we add a MSE loss on the predictions of h_f (weighted by K_h) to encourage h_f to be consistent with the true latent dynamics.

$$\mathcal{L}_\phi = -\mathop{\mathbb{E}}_{\mathcal{S}}\left[\log \frac{f(s_{t+k}, s_t)}{\sum_{s_j \in \mathcal{S}} f(s_j, s_k)}\right] + K_h \cdot \mathop{\mathbb{E}}_{\mathcal{Z}}\left[\sum_{i=0}^{k-1}(z_{t+1+i} - h_f(\widetilde{z}_{t+i}, a_{t+i}))^2\right] \tag{4}$$

Optimizing ϕ with Eq. 4, we intend to achieve locality of correlated states while shaping dynamics that are predictable by a model h_f. For environments conditioned on a task-specific context, we additionally feed c into the dynamics model, i.e. $h_f : \mathcal{Z}\times\mathcal{A}\times\mathcal{C}\rightarrow\mathcal{Z}$ and optimize ϕ considering the empirical expectation of the loss in Eq. 4 w.r.t. $\mathcal{C}$. The full encoder objective is provided in App. A.1

4.2 Conditional Transition Density

Due to the symmetry property of the previously defined CPC critic f, our embedding cannot fully capture causal relationships between states. We organize subsequent states close together but lose the information in which direction transitions were experienced, i.e the arrow of time. This knowledge of connectivity is yet required to rewire nodes in the search tree and indispensable for environments with asymmetric dynamics. To recover it, we approximate the conditional density $p(z_{t+1}|z_t, c)$ of latent transition $z_{t:t+1|c}$ using Noise-Contrastive Estimation (NCE) (Sec. 3). More specifically, we employ a parametric model $\psi : \mathcal{Z}\times\mathcal{Z}\times\mathcal{C} \rightarrow \mathbb{R}$ to represent the set of log density estimators given by the conditioning on the transition source node z_t and environment context c. We train ψ by maximizing the expectation of the NCE objectives $\mathcal{J}_{\text{NCE}}$ over the empirical distribution of all conditionings for the observed data, hence $\mathcal{J}_\psi = \mathbb{E}_{c\sim\mathcal{C}, z_t\sim\mathcal{Z}}\left[\mathcal{J}_{\text{NCE}}(c, z_t)\right]$. A discrete version of this strategy has previously been used in [35] for estimating the context-dependent probability of words in probabilistic language models.

The crux of NCE is to find a noise distribution p_n which tightly encloses the support of the unknown target distribution (Sec. 3), in our case $p(z_{t+1}|z_t, c)$. Our key insight is however that due to the contrastive training of ϕ, subsequent states are encouraged to be mapped close together in $\mathcal{Z}$. Hence, a simple choice for p_n is found by centering a multivariate Gaussian $\mathcal{N}(z_t, \Sigma)$ at each starting state z_t of a transition and setting the value of Σ empirically given the absolute lengths of all observed latent transitions in the data (Fig. 3).

One motivation of using NCE instead of other density estimation techniques is that it allows us to use powerful neural network function approximators. More importantly, the discriminative training procedure given p_n provides examples of outlier points for which our model is encouraged to assign low density. Extrapolating densities outside our data is an important aspect for correctly identifying unlikely transitions.

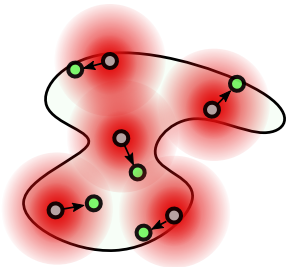


Figure 3: Illustration of positive pairs and noise for training the transition density model ψ via NCE. Gaussian noise distribution p_n in red, start states of transitions z_t in gray and end points of transitions z_{t+1} in green.

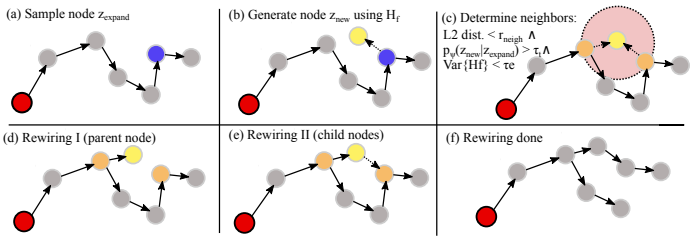


Figure 4: Tree growth and rewiring mechanism. Repeat: **(a)** select random node z_{expand} (blue) in the existing tree **(b)** create new node z_{new} (yellow) from z_{expand} using random action and dynamics H_f **(c)** find set of reachable nodes Z_{neigh} (yellow) in local neighborhood around z_{new} **(d)** change parent of z_{new} if there exist a node in Z_{neigh} for which z_{new} is reachable and which reduces the number of steps between z_{start} and z_{new} **(e), (f)** alter parent for every nodes in Z_{neigh} if transitioning from z_{new} is possible and reduces traveling cost.

4.3 Tree Expansion and Optimization

Due to the unboundedness of the latent space $\mathcal{Z} = \mathbb{R}^{dz}$, erroneous latent transitions during the search might result in endless exploration of undefined areas outside the latent distribution. Once the embedding is learned, we train a powerful dynamics ensemble estimator $H_f = \{h_f^1, .., h_f^m\}$ to reduce the number of detrimental transitions. Furthermore, this allows us to evaluate the ensemble's predictive uncertainty [25] and reject highly uncertain predictions during planning.

Given the encoder ϕ, forward dynamics H_f and transition density model ψ, all the building blocks that compose our planner are introduced. For every new planning query, we start by adding the initial latent state z_{start} to the tree and then alternate between expanding and optimizing the search tree. Fig. 4 illustrates the mechanisms behind this procedure.

Tree Expansion During the expansion step, we sample a node z_{expand} from the current tree and generate a new state z_{new} from it using H_f and a random action $a \sim \mathcal{A}$. The newly generated transition is discarded if the density estimated by ψ is below a threshold τ_t or the predictive uncertainty $\text{Var}(\{h_f^i, i = 1..m\})$ exceeds a threshold τ_e. If both tests are passed, we connect z_{new} to the tree and continue with the rewiring step.

Rewiring We optimize the search tree by rewiring edges locally around each newly added node if this reduces the traveling distance from the root. Firstly, we find the set of neighboring tree nodes Z_{neigh} within a ball of radius r_{neigh} around z_{new}. Next, we select the ones that are reachable from z_{new}, i.e. the associated transition density $\geq \tau_t$. For all the remaining nodes, including z_{expand}, we determine the number of steps to travel from z_{start} to z_{new}. The node associated with the minimum cost becomes the new parent node of z_{new}. For each node in Z_{neigh}, we then make z_{new} the parent node if such a transition is reachable and reduces the traveling cost from z_{start}. Note that this rewiring technique presents an adaptation of the one used in [21] where it yields asymptotically-optimal versions of sampling-based planners such as RRT. In stark contrast to [21], we employ learned connectivity and dynamics models and operate on a tree composed of neighboring latent encodings.

Our planner stops after n_{iter} expansion steps. Paths between the root and any other node are then obtained by backtracking upwards the tree. We return the path with the least traveling cost that ends within the vicinity of z_{goal} (ball of radius r_{goal}). If no such paths exist, we simply take the one with the least Euclidean distance to the goal.

Sampling Bias Sampling z_{expand} uniformly leads to slow expansion of the search tree. To encourage the planner to grow quickly into unexplored regions, we must bias the sampling towards less

represented areas in $\mathcal{Z}$. For each node, we assign a weight based on the inverse number of its reachable neighbors and sample z_{expand} from the weighted distribution with probability p_{sparse} and uniformly with p_{uniform}. Moreover, with p_{goal}, the node which is closest to the goal with respect to the Euclidean latent space distance is chosen. We found this heuristic to be effective at encouraging exploration towards the goal. Further details about sampling biases are provided in App. A.1.2.

4.4 Integration with Model-Predictive Control

To apply our planner to sequential navigation tasks, we integrate ELAST into a simple Model Predictive Control (MPC) scheme (Fig. 2) in which we replan after n_{replan} interactions. The controller queues the planned sequence of intermediate states to be achieved by a local control policy. For simplicity, we employ a simple one-step policy $\pi : \mathcal{Z} \times \mathcal{Z} \times \mathcal{C} \to \mathcal{A}$ which we train supervised with $\mathcal{L}_\pi = \mathbb{E}_{\mathcal{Z}} \left[(a_t - \pi(z_t, z_{t+1}))^2 \right]$.

5 Experiments

In this section, we evaluate the effectiveness of ELAST by embedding it in an MPC loop (section 4.4) and measuring its performance in terms of success rates on challenging downstream control tasks with long time horizons and visual observations. We also investigate how the planner's hyperparameters affect the exploration behavior and the quality of the computed solutions.

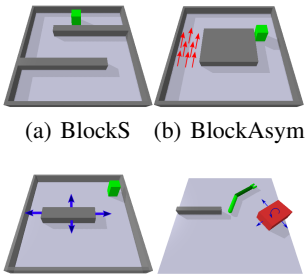
5.1 Experimental Setup

Baselines To put the performance of ELAST into perspective, we compare it to the following baselines.

- **PlaNet [13]** A popular latent planning baseline based on shooting-based trajectory optimization via the Cross-Entropy Method [3].
- **HTM [32]** Hallucinative Topological Memory is an image-based planning method mainly designed for context-conditioned tasks. HTM plans on a graph of visual states generated by sampling observations from a Variational Autoencoder [23].
- **VGCBC** Visual goal-conditioned behavioral cloning from images
- **CPC-CEM** CEM [3] applied under the CPC embedding
- **CPC-Coll.** Collocation applied under under the CPC embedding
- **CPC-GCBC** Behavioral cloning under the latent CPC embedding

We futher implemented LatCo [40] which did not produce sensible results in our settings and is therefore excluded in the experimental evaluation. Further information and other details about baselines are provided in App. C.

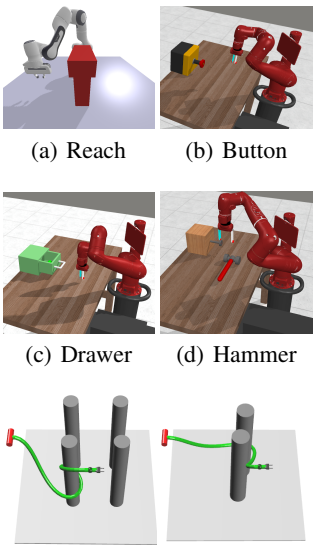
Control Tasks To better disentangle the effect of different environment characteristics on performance of the planning, several toy environments were designed as illustrated in Fig. 5. In *BlockS*, a block agent must navigate through an S-shaped corridor representing a long-horizon and geometrically non-convex setting. *BlockAsym* introduces asymmetric transition dynamics through a unidirectional stream on one side of the workspace. Parameterized environments conditioned on the position of obstacles are studied in both *BlockParam* and *PlanarArm*. The simplicity of the planar workspace allows us to interpret planned latent paths visually for instance by looking at the resulting 2D Isomap [47] embeddings. Moreover, we tested our method on the visually more challenging robot control tasks in 6. This includes a robot reaching task (Fig. 6 a) and three manipulation environments (Fig. 6 b-d) adapted from the metaworld benchmark [55] (used in [40]). Moreover, we implemented both a static and parameterized version of a cable navigation task (Fig. 6 e and f).



(a) BlockS (b) BlockAsym

(c) BlockParam (d) PlanarArm

Figure 5: Planar Environments



(a) Reach (b) Button

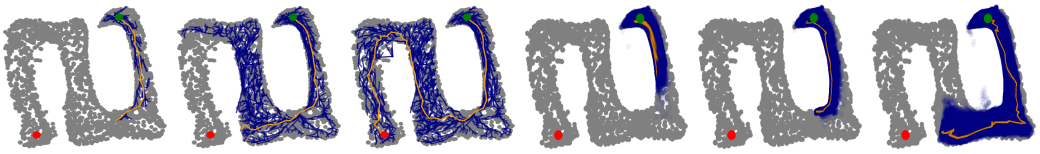
(c) Drawer (d) Hammer

(e) Cable (f) CableParam

Figure 6: Visually-complex tasks

Table 1: Success rates (%) for test scenarios averaged over three independent training runs

METHOD	BLOCKS MEDIUM	BLOCKS HARD	BLOCK ASYM	BLOCK PARAM	PLANAR ARM	REACH	BUTTON	DRAWER	HAMMER	CABLE	CABLE PARAM
ELAST	99 ± 1	96 ± 3	100 ± 0	84 ± 2	88 ± 1	96 ± 2	88 ± 5	84 ± 12	22 ± 2	95 ± 2	79 ± 11
CPC-CEM	53 ± 2	9 ± 4	21 ± 7	68 ± 3	64 ± 1	68 ± 6	1 ± 0	32 ± 2	0 ± 0	29 ± 4	17 ± 3
CPC-COLL.	15 ± 1	0 ± 0	7 ± 5	3 ± 2	21 ± 7	3 ± 2	1 ± 1	44 ± 2	0 ± 0	5 ± 1	26 ± 1
CPC-GCBC	20 ± 2	0 ± 0	0 ± 0	29 ± 4	57 ± 2	31 ± 2	20 ± 14	2 ± 1	0 ± 0	26 ± 5	20 ± 1
PLANET	17 ± 1	0 ± 0	0 ± 0	11 ± 0	42 ± 12	6 ± 1	18 ± 3	34 ± 9	0 ± 0	5 ± 1	38 ± 5
HTM	7 ± 3	0 ± 0	15 ± 8	28 ± 11	43 ± 7	5 ± 6	45 ± 4	3 ± 2	0 ± 0	19 ± 3	30 ± 2
VGCBC	25 ± 8	0 ± 0	0 ± 0	35 ± 2	56 ± 1	37 ± 1	81 ± 9	48 ± 3	13 ± 2	1 ± 1	1 ± 1
RANDOM	8 ± 2	0 ± 0	0 ± 0	1 ± 1	2 ± 1	0 ± 0	1 ± 0	4 ± 2	0 ± 0	8 ± 3	1 ± 1



(a) n_{iter}=500 (b) n_{iter}=5000 (c) n_{iter}=7500 (d) $(5, 20, 500)$ (e) $(10, 20, 500)$ (f) $(10, 50, 1000)$

Figure 7: Exploration behavior of ELAST and CEM illustrated within 2D Isomap embeddings for the *BlockS* task. The goal (red) and start (green) lie on different ends of the corridor. (a)-(c): ELAST: search tree (blue) and best path (orange) at different iterations n_{iter}. (d)-(f) row: CEM visited area (blue) and final population (orange) for different (I, H, K) with iterations I, horizon H and K candidates.

For all environments, we study goal-reaching tasks by providing an observation s_{goal} of the desired target configuration. We form current observations s_t by stacking three subsequent RGB frames of resolution 64×64. In parameterized environments, we obtain context observations c by taking the latent encodings given by a convolutional autoencoder which is trained separately on context training images. We evaluate the success rates of trajectories on 100 unseen scenarios for each task. Detailed information about implementation of the environment and experimental evaluation are provided in App. A, B .

5.2 Experimental Results

Does ELAST outperform existing methods on long-horizon visual goal-reaching tasks? The results of our benchmark evaluation are presented in Table 1. We observe that ELAST significantly outperforms existing baselines on downstream control tasks in terms of achieved success rates. This is both true for highly non-convex settings such as the *BlockS-hard* task and visually complex manipulation. The ablation comparison with CPC-CEM,CPC-Coll. and CPC-GCBC shows that it is not alone the choice of representation but indeed our planner that contributes to the success of ELAST. Fig. 7 depicts the exploration behavior in the *BlockS* environment for ELAST and CPC-CEM. It presents the Isomap projections of the explored latent areas for different number of generated samples during planning. Our method expands more quickly into the latent space and connects the goal to the tree with less than 7500 generated states. CPC-CEM however did not reach deep enough into the space to compute a valid solution path. Even with a substantially larger number of evaluated states (I=10, H=50,K=100 is equivalent to 10^5 generated samples), it does not achieve the same level of space coverage. HTM did not produce robust solutions in comparison to our approach. During evaluation, it was observed that the generated images were indeed visually sound but few remaining image artifacts introduced shortcut connections in the graph which often rendered entire paths infeasible. This result underpins our motivation to pursue planning with abstract and lower-dimensional representations of the inputs to avoid the challenge of maintaining accurate high-dimensional observations. Further visualizations of successful latent paths for other environments are provided in App. D.9. While our method performs best among all baselines for the *Hammer* environment, we only achieved 22% success for this task. We account this to the difficulty of planning through narrow passages in higher-dimensional spaces introduced by the required grasping task (See App. D.10). Further, we noticed that the agent often failed to properly grasp the hammer which in the future could be improved using more sophisticated RL policies for our local controller π.

How does our NCE transition density model impact the performance of the planner? To answer this question, we tested ELAST in the *BlockS* and *Cable* environments without rejecting transition during planning. We observed a significant drop in performance for both tasks from $96 \pm 3\%$ to $51 \pm 4\%$ in *BlockS-hard*, respectively from $95 \pm 2\%$ to $25 \pm 5\%$ in the *Cable* task. Fig. 8a shows the predicted neighbor latent states with respect to a node in the unidirectional passage in the *BlockAsym* environment. As shown, our transition density model correctly determines state connectivity even under highly asymmetric transition dynamics. Without transition rejection enabled, the performance on this task dropped to $73 \pm 5\%$ as a result of infeasible shortcut paths against the direction of the stream. The results indicate that our model ψ generates useful density estimates for latent transitions and is indeed crucial for the performance of ELAST.

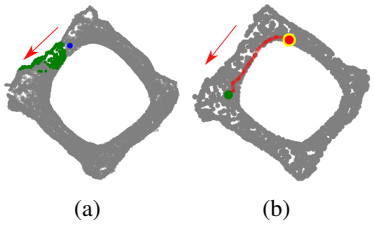


Figure 8: (a): Predicted neighboring states (green) for node (blue) within the unidirectional passage in *BlockAsym* (projected with Isomap). Obtained by thresholding estimated transition densities (c): Infeasible path from an initial state (green) towards a goal (red) when transition rejection is disabled

Does the rewiring affect the quality of generated paths? Our tree rewiring module (Sec. 4.3) encourages smooth and near-optimal paths which is crucial to finish the control tasks within the allowed contingent of environment steps. We evaluated ELAST on the *BlockS-hard* and the *Button* tasks without optimizing the tree. The success rates decreased drastically from $96 \pm 3\%$ to $40 \pm 4\%$ in *BlockS-hard* and from $88 \pm 5\%$ to $50 \pm 2\%$ in the *Button* environment. For the remaining successful episodes, we observed an increase of average trajectory length from 36 ± 1 to 93 ± 2 and from 38 ± 6 to 72 ± 5 steps in the *BlockS-hard*, respectively the *Button* task. Our results confirm the necessity of path rewiring to reduce trajectory lengths and thereby improve the success of our method.

Choice of sampling bias and planning hyperparameters The exploration behavior of ELAST is largely determined by the weighting of the sampling biases. We found that values of $p_{\text{uniform}} = 0.2$, p_{sparse}=0.78 and p_{goal}=0.02 produce sample-efficient and goal directed exploration in all experiments. Depending on the difficulty of the tasks we found that planning iterations between n_{iter}=2500 and n_{iter}=7500 were sufficient to compute feasible paths for all environments. A detailed discussion of planning hyperparameters and further experimental results are provided in App. A.1.2 and D.

6 Limitations and Future Work

Regardless the node sampling bias introduced in Sec 4.3, our planner expands the search tree using uniform action selection. While sufficient exploration is important, this can lead to the tree branching out to areas of the latent space which are far from the goal and later do not contribute to the solution. To improve the sample-efficiency of ELAST, sampling heuristics could be utilized to bias node sampling and further guide the planner through narrow passages. Similarly, we could integrate better local policies to guide the action-selection and further improve the performance of the local controller to reach the next waypoint in the planned path. Related work has applied deep generative networks to learn state sampling for classical robot motion planning [20]. In the scope of this work, we focus on planning in long-horizon goal-reaching scenarios. Similar to [13, 42, 40], another direction is to integrate ELAST into the reinforcement learning framework and instead consider the predicted accumulated discounted reward along the tree during planning. This would enable discovery of far-distant and high rewarding areas instead of just single goals.

7 Conclusion

We introduced ELAST, a method for solving high-dimensional and long-horizon planning tasks in latent spaces. By merging ideas from classical sampling-based motion planning and self-supervised contrastive learning, our method initiates a search tree which is bound to the estimated support of the latent manifold. We demonstrated its effectiveness over existing latent planning methods on challenging and visually complex planning environments and hope to inspire others to investigate new ways to combine concepts from classical planning and machine learning.

References

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

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[3] 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.

[4] brian ichter, Pierre Sermanet, and Corey Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. In *5th Annual Conference on Robot Learning*, 2021.

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

[6] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control. *arXiv e-prints*, page arXiv:1812.00568, December 2018.

[7] 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.

[8] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[9] Robert Gieselmann and Florian T. Pokorny. Planning-augmented hierarchical reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):5097–5104, 2021.

[10] 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.

[11] 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.

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

[13] 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.

[14] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. In *Advances in Neural Information Processing Systems*, volume 34, pages 26661–26673. Curran Associates, Inc., 2021.

[15] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and Remi Munos. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[17] 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.

[18] Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

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

[20] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094, 2018.

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

[22] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[24] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[25] 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.

[26] Michael Laskin, Scott Emmons, Ajay Jain, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In *Advances in Neural Information Processing Systems*, 2020.

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

[28] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[29] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[30] 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.

[31] 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.

[32] 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.

[33] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

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

[35] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models, 2012.

[36] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1101–1112. PMLR, 30 Oct–01 Nov 2020.

[37] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020.

[38] 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.

[39] Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8130–8139. PMLR, 18–24 Jul 2021.

[40] Oleh Rybkin, Chuning Zhu, Anusha Nagabandi, Kostas Daniilidis, Igor Mordatch, and Sergey Levine. Model-based reinforcement learning via latent-space collocation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9190–9201. PMLR, 18–24 Jul 2021.

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

[42] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *ICML*, 2020.

[43] 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.

[44] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[45] 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.

[46] Russ Tedrake. *Underactuated Robotics*. 2022.

[47] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[48] Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2021.

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

[50] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[51] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[52] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016.

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

[54] Lin Yen-Chen, Maria Bauza, and Phillip Isola. Experience-embedded visual foresight. In *Conference on Robot Learning*, 2019.

[55] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 30 Oct–01 Nov 2020.

[56] 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.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 6 and App. D.10.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our method describes a general planning algorithms for high-dimensional data under unknown dynamics. We do not foresee any immediate societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemented code repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See App. A.1.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See App. A.1.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]