Comparing Reconstruction- and Contrastive-based Models for Visual Task Planning

Constantinos Chamzas*[1], Martina Lippi*[2], Michael C. Welle*[3],
Anastasia Varava[3], Lydia E. Kavraki[1], and Danica Kragic[3]

***Abstract*— Learning state representations enables robotic planning directly from raw observations such as images. Most methods learn state representations by utilizing losses based on the reconstruction of the raw observations from a lower-dimensional latent space. The similarity between observations in the space of images is often assumed and used as a proxy for estimating similarity between the underlying states of the system. However, observations commonly contain task-irrelevant factors of variation which are nonetheless important for reconstruction, such as varying lighting and different camera viewpoints. In this work, we define relevant evaluation metrics and perform a thorough study of different loss functions for state representation learning. We show that models exploiting task priors, such as Siamese networks with a simple contrastive loss, outperform reconstruction-based representations in visual task planning.**

I. Introduction

Learning of low-dimensional state representations from high-dimensional observations such as images have gained significant attention in robotics [1], [2]. For complex manipulation planning tasks, this approach is a viable alternative since analytic approaches may be computationally expensive or impossible to define. Existing approaches are generally based on an implicit assumption that *similar observations, close in the image space, correspond to similar system states*. However, the same underlying state may be related to very different observations due to other factors of variation, such as different views or background of the scene, see Fig. 1. This is especially true in *task* planning, which we focus on, where states are typically discrete and their observations may be captured in very different time intervals, leading to the natural occurrence of task irrelevant factors of variation. Similar considerations also hold for task and motion planning (TAMP) settings [3].

To address this, it is crucial to identify the *task-relevant* factors of variation. A step in this direction is done by [4], where an agent interacts with the environment and tries to disentangle the controllable factors of variation. However, if data is being collected in realistic scenarios, irrelevant factors of variation may occur that are difficult to control.

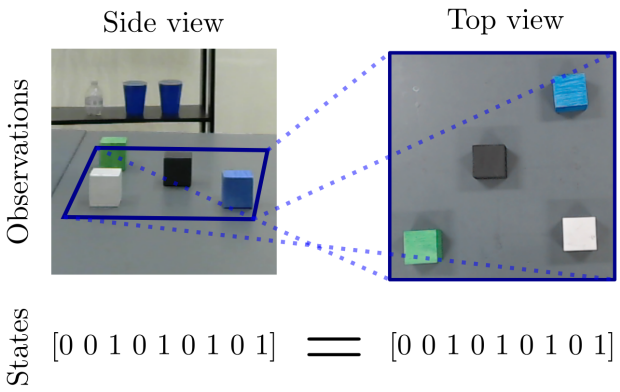

Fig. 1: Examples of visually different observations (different views) of the same state (arrangement of the boxes).

Although several solutions exist in literature, a unified analysis of representation losses and their influence to the performance of learned representations for high-level visual task planning is currently missing. In this work, we perform a systematic comparison of different representation learning methods which can possibly leverage task priors in quasi-static tasks. To this aim, we also design and collect datasets where the underlying states of the system do not uniquely correspond to observations (images). We study a box manipulation task on a real robotic system as well as a simulated shelf arrangement task. In all tasks, different task-irrelevant factors, such as different viewpoints of the same scene or "distractor" objects, are present. Our work makes the following contributions: *i)* We introduce evaluation metrics and provide a systematic study for assessing the effect of different loss functions on state representation. Robotic tasks on both real hardware and simulation are analyzed. *ii)* We examine a simple data augmentation procedure for contrastive-based models. *iii)* We show how task priors in contrastive-based models combined with simple data augmentations can lead to the best performance in visual task planning with task-irrelevant factors of variation and demonstrate the performance of the best derived representations on a real-world robotic task. *iv)* We create and distribute datasets for comparing state representation models[1].

II. Related Work

State representation learning from high-dimensional data has been successfully used in a variety of robotic tasks.

*Contributed equally (listed in alphabetical order)

This project has been supported in part by NSF GRFP 1842494 (CC) and NSF 2008720 (LEK)

[1] Rice University, 6100 Main St, Houston, TX 77005, USA chamzas, kavraki@rice.edu
[2]Roma Tre University, Via Ostiense, 133B, 00154 Roma RM, Italy martina.lippi@uniroma3.it
[3]KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden mwelle, varava, dani@kth.se

[1] https://state-representation.github.io/web/

As shown in Table I, the used loss functions are usually a combination of the reconstruction, Kullback–Leibler (KL)-divergence, and contrastive loss functions. A common approach to use learned state representations is through learned forward dynamic models as in [5], [6], [7], [8]. These dynamic models predict future observations (images) and are trained to minimize the pixel distance between the observed image and the decoded predicted observation. Among these works, [5] also exploits a KL loss to regularize the latent space. Future rewards and actions are predicted instead in [9], and the image reconstruction loss is solely used to regularize. Since in many cases predicting full images is not practical, some approaches attempt to remove task-irrelevant information from the predicted images. In [10], the residual of goal and the current state is reconstructed which contains more relevant information comparing to a raw image. Similarly, in [11] images are transformed through specialized layers that enhance spatial features, such as object locations. Learned representations leveraging reconstruction loss have also been used in specific robotic applications, such as [12] for fabric manipulation and [13] for pendulum swing up.

As shown in Table I, all the aforementioned methods rely on the reconstruction loss. However, in many real scenarios, full images might contain redundant information, making the reconstruction loss not applicable. Inspired by the revival of contrastive methods in computer vision [14], some recent works rely on contrastive losses to learn efficient state representations. The works in [2], [15], [16] augment pixel frames through transformations and use a forward momentum encoder to generate positive and negative examples. These examples are then exploited to learn state representations directly in the latent space without the need for a decoder. In [17], a purely contrastive loss is used to learn robotic states from video demonstrations where states that are temporally close are considered similar. In addition, the authors of [18] remove task-irrelevant information by adding distractors during simulation and considering such states similar in their contrastive loss formulation. Contrastive-like losses have also been formulated using task or robotic priors [19] such as slowness [20]. The latter has been applied in reinforcement learning [21] with visual observations, and humanoid robotics state representation [22]. A no-action/action prior was also used in our previous work [23], which was used to formulate a combined reconstruction, KL, and contrastive loss. Here, we leverage the same task prior of [23] as explained in Sec. III.

The vast majority of the aforementioned methods are concerned with continuous control tasks, whereas in this work we are focusing on quasi-static states tailored towards long-horizon high-level planning [1]. In detail, we take representative models employing different loss functions and perform a thorough study by analyzing their performance in robotic planning tasks with and without task priors. Such discrete state representations have been learned in literature by object-centric or compositional models like [16], [24], however we do not assume any structural relations between observations.

Related works	Recon.	KL	Contr.
[6], [7], [8], [9], [12], [13]	✓		
[5], [10], [11], [25]	✓	✓	
[2], [15], [16], [17], [18]			✓
[23]	✓	✓	✓

TABLE I: Overview of loss functions (reconstruction, KL divergence and contrastive) used by state-of-the-art methods.

III. Problem Formulation

Our objective is to define appropriate state representations for visual task planning given high-dimensional observations provided as images. Let $\mathcal{O}$ be the observation space and $\mathcal{Z}$ be a low-dimensional state space, also referred to as latent space. The goal is to define a mapping function $\xi : \mathcal{O} \rightarrow \mathcal{Z}$ which extracts the low-dimensional representation $z \in \mathcal{Z}$ given a high-dimensional observation $o \in \mathcal{O}$. We consider that task-irrelevant factors can be present in the observations which cause them to be possibly *visually dissimilar* even if they contain the same underlying states.

An ideal mapping function ξ^* should be able to perfectly capture the underlying states of the system despite possible task-irrelevant factors. This means that, given two observations o_i and o_j containing the same state, it holds $\xi^*(o_i) = \xi^*(o_j)$, i.e., they are mapped into the same latent encoding. We aim to understand how to model ξ such that it is as close as possible to ξ^* when task-irrelevant factors are present in $\mathcal{O}$.

Although a perfect mapping ξ^* might not be achievable, a good approximation should be able to properly structure the latent space such that encodings associated with the same states are close by, while encodings that are associated with different states are well separated in the latent space. This implies that the encodings should be *clustered* in the latent space such that each cluster is associated with a possible underlying state of the system. Note that if such clustering is achieved, task planning can be easily solved by properly connecting the clusters when an action is allowed between the respective states of the system. Therefore, better mapping results in improved clusters and requires an easier planning algorithm. An illustrative example is provided in Fig. 2, where three latent spaces $\mathcal{Z}^*, \mathcal{Z}^1, \mathcal{Z}^2$, obtained with different mapping functions, ξ^*, ξ^1, ξ^2, are shown. Considering that observations (top row) in the same colored box contain the same underlying state, it can be observed that *i)* the latent space $\mathcal{Z}^*$ (bottom left) is optimal since observations containing the same states are mapped exactly into the same latent encoding, *ii)* a sub-optimal latent space $\mathcal{Z}^1$ (bottom middle) is obtained since the latent encodings are properly clustered according to the states of the system, *iii)* a very hard-to-use latent space $\mathcal{Z}^2$ (bottom right) is obtained where the encodings are not structured according to the states.

Training Dataset: To model the mapping function, we assume task priors are available to build the training dataset. In detail, a training dataset $\mathcal{T}_o$ is composed of tuples (o_i, o_j, s) where $o_i, o_j \in \mathcal{O}$ are observations of the system, and $s \in \{0, 1\}$ is a signal, obtained from task priors, specifying

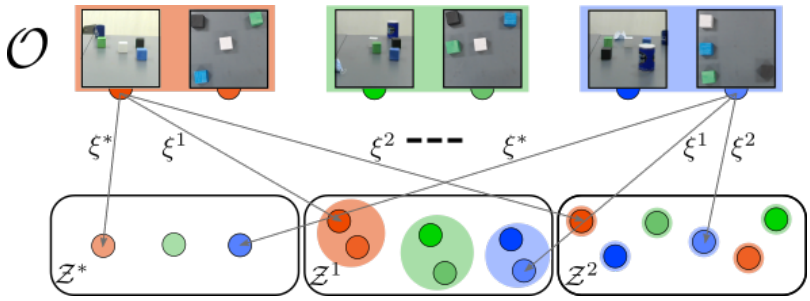

Fig. 2: Examples of mapping functions ξ^*, ξ^1, ξ^2 (arrows) from observation space $\mathcal{O}$ (top row) to latent spaces $\mathcal{Z}^*, \mathcal{Z}^1, \mathcal{Z}^2$ (bottom row). Boxes arrangement represents the system state and images marked with variations of the same color contain the same state.

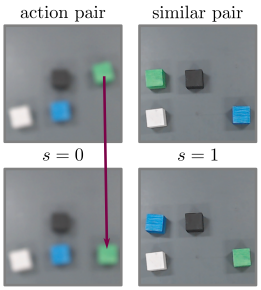

Fig. 3: Example of action (left) and similar (right) pairs. We consider the boxes interchangeable (only the resulting arrangement matters).

whether the two observations are *similar* ($s = 1$), i.e., they correspond to the same state and $\xi^*(o_i) = \xi^*(o_j)$, or whether an *action* occurred between them ($s = 0$), i.e., they represent consecutive states, implying that o_i and o_j are dissimilar and $\xi^*(o_i) \neq \xi^*(o_j)$. An action represent any high-level operation as in [1], e.g., pick and place, pushing, and pouring operations. We refer to the tuple as a *similar pair* when $s = 1$, and as an *action pair* when $s = 0$. In addition, the encoded training dataset composed of tuples (z_i, z_j, s), with $z_i = \xi(o_i)$ and $z_j = \xi(o_j)$, is denoted by $\mathcal{T}_z$.

Note that in both similar and action pairs task-irrelevant factors can change in the observations o_i, o_j, i.e., it generally holds $o_i \neq o_j$, while task-relevant factors only change through actions in action pairs. Moreover, no knowledge of the underlying states of the training observations is assumed. Examples of action and similar pairs are shown in Fig. 3 for a box manipulation task (with interchangeable boxes), as detailed in Sec. VII. The training dataset can be generally collected in self-supervised manner. Indeed, action pairs can be obtained by randomly performing high-level actions with the environment similar to [1] and recording the respective consecutive observations. Regarding similar pairs, they can be obtained, for example, by recording observations in the tuple with a certain time separation, leading to the occurrence of different lighting conditions and/or the presence of further irrelevant objects in the scene, or, as in our experiments, by swapping objects if they are interchangable for the task.

Data Augmentation: Inspired by the training procedure in [26], we consider a synthetic procedure to generate an additional training dataset $\overline{\mathcal{T}}_o$ from $\mathcal{T}_o$. Let $\mathcal{O}_T$ be the set of all observations in $\mathcal{T}_o$. The basic idea is that by *randomly* sampling pairs of observations in the dataset, they will *likely* be dissimilar. Therefore, $\overline{\mathcal{T}}_o$ is first initialized to $\mathcal{T}_o$. Then, for each similar pair $(o_i, o_j, s = 1) \in \mathcal{T}_o$, we randomly sample n observations $\{o_1^s, ..., o_n^s\} \subset \mathcal{O}_T$ in the dataset and define the tuples $(o_i, o_k^s, s = 0)$, $k = 1, .., n$, which are added to $\overline{\mathcal{T}}_o$. In this way, for each similar pair, n novel tuples are introduced in $\overline{\mathcal{T}}_o$ with respect to $\mathcal{T}_o$. We experimentally validate that this procedure allows to improve the latent mapping despite possible erroneous tuples in $\overline{\mathcal{T}}_o$, i.e., novel tuples for which it holds $\xi^*(o_i) = \xi^*(o_k^s)$.

IV. Latent Mapping Modeling

We employ and compare different unsupervised and prior-based, i.e., using the similarity signal s, models as follows.
i) The classic *Principal Component Analysis (PCA)* method [27] is used as an unsupervised baseline method. It obtains the latent mapping by finding the eigenvectors with the highest eigenvalue from the dataset covariance matrix.
ii) Auto-Encoder (AE) [28] is considered as another unsupervised approach. AE is composed of an encoder and a decoder network trained jointly to minimize the Mean Squared Error (MSE) between the input o and decoded output $\tilde{o}$:

$$\mathcal{L}_{ae}(o) = (o - \tilde{o})^2.$$

iii) A standard *β-Variational Auto-Encoder (VAE)* [29] is considered as an additional unsupervised model. Similarly to the AE, the β-VAE consists of an encoder and a decoder network which are jointly trained to embody the approximate posterior distribution $q(z|o)$ and the likelihood function $p(o|z)$ providing generative capabilities. The following loss function is minimized:

$$\mathcal{L}_{\beta-vae}(o) = E_{z\sim q(z|o)}[\log p(o|z)] + \beta \cdot D_{KL}\left(q(z|o)||p(z)\right)$$

with z the latent variable, $p(z)$ the prior distribution realized as a standard normal distribution and $D_{KL}(\cdot)$ the KL divergence.
iv) The similarity signal can be exploited through a Pairwise Contrastive (PC) loss [23], encouraging the encodings of action pairs to be larger than a certain distance while minimizing the distance between similar pairs. This loss is used to augment the standard AE loss as follows [30]:

$$\mathcal{L}_{pc-ae}(o_i, o_j, s) = \frac{1}{2}\left(\mathcal{L}_{ae}(o_i) + \mathcal{L}_{ae}(o_j)\right) + \alpha \mathcal{L}_{pc}(o_i, o_j, s)$$

with α a hyperparameter and $\mathcal{L}_{pc}(o_i, o_j, s)$ defined as

$$\mathcal{L}_{pc}(o_i, o_j, s) = \begin{cases} \max(0, d_m - ||z_i - z_j||_1^2) & \text{if } s = 0 \\ ||z_i - z_j||_1^2 & \text{if } s = 1 \end{cases} \quad (1)$$

where d_m is a hyperparameter denoting the minimum distance that is encouraged between encodings of the action pairs. We denote the resulting model as *PC-AE*.
v) Similarly to the PC-AE, the task priors can also be used to combine the β-VAE loss with the PC loss, leading to the

Model	Recon. loss	KL loss	Contr. loss
PCA			
AE	✓		
β-VAE	✓	✓	
PC-AE	✓		✓
PC-VAE	✓	✓	✓
PC-Siamese			✓
CE-Siamese			✓

TABLE II: Summary of the considered models with respect to their loss functions.

following loss function [23]

$$\mathcal{L}_{pc-vae}(o_i, o_j, s) = \frac{1}{2}\left(\mathcal{L}_{\beta-vae}(o_i) + \mathcal{L}_{\beta-vae}(o_j)\right) + \gamma \mathcal{L}_{pc}(o_i, o_j, s)$$

with γ a hyperparameter. We denote this model as *PC-VAE*.
vi) A pure contrastive-based model is then considered which is a Siamese network with pairwise contrastive loss [31], referred to as *PC-Siamese*. This model structures the latent space such that it minimizes the pairwise distance between similar pairs and increases it between dissimilar pairs. As dissimilar pairs, the action pairs are used ($s = 0$). This model is based on the sole PC loss $\mathcal{L}_{pc}(o_i, o_j, s)$ in (1), i.e., it only relies on the similarity signal while no use of reconstruction loss is made.
vii) A further Siamese network model is considered with different contrastive loss function. In particular, the following normalized temperature-scaled Cross Entropy (CE) loss [26], [32] is leveraged which minimizes the cross-entropy between similar pairs using the cosine similarity: This model relies on the following normalized temperature-scaled cross-entropy loss [26], [32]:

$$\mathcal{L}_{ce}(o_i, o_j) = -\log\left(\frac{e^{(\mathrm{sim}(z_i, z_j)/\tau)}}{\sum_{k=1}^{2N} \mathbb{1}_{k\neq i} e^{(\mathrm{sim}(z_i, z_k)/\tau)}}\right) \quad (2)$$

where $\mathrm{sim}(u, v) = u^\top v/\|u\|\|v\|$ is the cosine similarity, $\mathbb{1}$ is the indicator function, τ is the temperature parameter and N is the number of similar pairs that are given in each batch. The resulting model is denoted by *CE-Siamese*. We use the training procedure in [26] where, for every similar pair, the rest $2(N-1)$ examples are considered dissimilar as in (2).
Models Summary: As summarized in Table II, the considered models allows to cover a wide range of losses. The PCA model is employed as a simple baseline to show that the tasks at hand have adequate complexity and cannot be solved with a PCA model. The AE and β-VAE models are mostly based on the reconstruction loss and therefore implicitly assume that a visible change in the observations corresponds to a state change. The latter models are then augmented in the PC-AE and PC-VAE models with a pairwise contrastive loss which exploits the task priors ameliorating the visual similarity assumption. In addition, PC-Siamese and CE-Siamese only rely on a contrastive loss without generative capabilities. However, the latter are often not required for downstream tasks. For the sake of completeness, in Sec. VIII, we also compare to the case in which no model is used, and raw observations are directly exploited.

V. Latent Planning

As we are interested in ultimately use learned representations for task planning, we leverage planning in the latent space as a quality measure itself of the representation, as detailed in the following section. We resort to our latent space planning method from [23] that builds a graph structure in the latent space, called Latent Space Roadmap (LSR). Algorithm 1 shows a high level description of the LSR building procedure.

Algorithm 1 Adapted LSR building [23]

Require: Dataset $\mathcal{T}_z$, min cluster size m
$\mathcal{G}$ = build-reference-graph($\mathcal{T}_z$) # Phase 1
$\mathcal{C}_z$ = HDBSCAN-clustering($\mathcal{T}_z$, m) # Phase 2
LSR = build-LSR($\mathcal{G}$, $\mathcal{C}_z$) # Phase 3
return LSR

The basic idea is to first build a reference graph using the encodings of action and similar pairs in $\mathcal{T}_z$ (Phase 1), i.e., nodes are created for each encoding and they are connected in case of action pairs. Next, in Phase 2, the latent space is clustered. We substitute the ε-clustering used in [23] with the HDBSCAN [33] which only requires the minimum samples per cluster m to be set. The LSR is then built in Phase 3 where each cluster is represented by a node that embodies the underlying state and clusters are connected through edges if they are one action apart, i.e., they contain encodings of action pairs. To use the LSR for planning, we first encode the start and goal observations with the model of interest and then select the respective closest nodes in the LSR as start and goal nodes of the path. Finally, we find the shortest paths from the start node to the goal one. Note that the objective of the planning is to produce a sequence of *actions* that lead from start to goal states. No decoded images are then needed and the LSR can be built in the latent space generated by any model in Sec. IV.

VI. Representation Evaluation Metrics

To evaluate the performance of the different latent mapping models, we propose two types of metrics. First, as stated in Sec. III, the structure of the latent space can be assessed through *clustering*, i.e., a good latent space should be easy to cluster. Second, the latent space should be suitable for task planning - a good latent space should result in easier planning. Thus, we also evaluate the *planning* performance of learned representations.

A. Clustering metrics

Homogeneity & Completeness: Given the ground truth states, the homogeneity score [34], denoted by h_c, measures the purity of the created clusters, i.e., that each cluster contains elements from the same state. Completeness, denoted by c_c, measures the preservation of the states in clusters, i.e., that each state is assigned to the same cluster. Both the metrics have range $[0, 1]$, with 1 being the best value. Assigning all elements in different clusters would results in

$h_c = 1$ and $c_c = 0$, while assigning all elements in the same cluster would results in $h_c = 0$ and $c_c = 1$. These quantities are calculated based on cross-entropy as formulated in [34].

Mean silhouette coefficient: The silhouette coefficient [35], denoted by s_c^i, is defined for each sample i and, in contrast to the previous metrics, does not rely on ground truth labels. Let d_{intra}^i be the mean distance between sample i and all the other points in the same cluster and let $d_{closest}^i$ be the mean distance between sample i and all other points in the closest cluster. The silhouette coefficient for sample i is defined as:

$$s_c^i = \frac{(d_{closest}^i - d_{intra}^i)}{\max(d_{intra}^i, d_{closest}^i)}$$

which can assume values in $[-1, 1]$, with higher values indicating dense and well-separated clusters. We report the mean silhouette coefficient s_c over all samples.

B. Planning Evaluation

To assess the planning performance achieved through the LSR, we evaluate both graph structure and the obtained start to goal paths. We define the *true* representative state for each node in the LSR as the state that is contained the most. The following metrics are considered:

Number of Nodes: It is the number of nodes in the LSR and is denoted by $|\mathcal{V}|$. This number should ideally be equal to the number of possible underlying states of the system.

Number of Edges: It represents the number of edges that are built between nodes in the LSR and is denoted by $|\mathcal{E}|$. In the case of optimal latent mapping and graph, the number of edges should be equal to the number of possible transitions between states of the system.

Correctness of the Edges: It is denoted by c_e and quantifies how many nodes are improperly connected in the LSR. In detail, it is defined as the number of *legal* edges, i.e., the edges associated to allowed state transitions according to the task rules, divided by the total number of edges. This score has range $[0, 1]$, with 1 being the best value.

Path Metrics: To evaluate the latent planning capabilities, we evaluate the correctness of the shortest paths between random novel start and goal observations (taken from holdout datasets). We consider 1000 different start and goal observations and evaluate the percentage that all found paths are correct, denoted by *% all*, and the percentage that at least one path is correct, denoted by *% any*.

VII. Validation Setting

Two tasks are considered: a box manipulation task on a real robotic system, and a simulated shelf arrangement, in Unity [36] environment. An additional simulated box stacking task can be found in our preliminary workshop paper [37] and is not reported here for the sake of brevity. It is worth highlighting that the goal of this work is not to solve these tasks in an innovative manner, but rather to gain general insights that can be transferred to cases where a determination of the exact underlying states is not possible.

In each task, the *task-relevant* objects are interchangeable – i.e., swapping two objects results in the same state. Their arrangement in the scene specifies the underlying state of the system. Other objects that are irrelevant for the task, referred to as *distractor* objects, can be present in the observations. The objective of all tasks is to plan a sequence of states to arrange the *relevant* objects according to a goal observation. Transitions between states – i.e., actions, can be then retrieved through the LSR [23]. All datasets are available on the website[1].

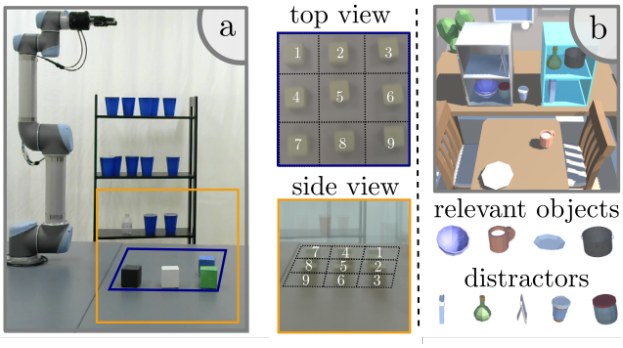

Fig. 4: **a)** Box manipulation dataset with two viewpoints. **b)** Shelf arrangement dataset with the task relevant objects (top object row) and the five distractor objects (bottom object row).

Box Manipulation: The setup of this real-world case study is shown in Fig. 4a). The task is composed of four interchangeable boxes, and each box can only move to adjacent tiles in a 3×3 grid. The robot is tasked with moving the boxes to the state of the goal image. This task has 126 possible states with 420 allowed state transitions. Two different viewpoints are considered to capture the scene and three datasets are built as follows: *i)* $\mathcal{BM}_s$, where all the observations are taken from the side view (in orange in Fig. 4), *ii)* $\mathcal{BM}_t$, where the observations are only taken from the top view (in blue in Fig. 4), and *iii)* $\mathcal{BM}_{st}$, where views are randomly picked from the side or top view. Images have naturally occurring task-irrelevant factors such as distractor objects changing in the background (side view), as well as out-of-focus images. In Fig. 4a) the mean image of all training images for side and top view are depicted. In the following, we report the considered self-supervised data collection procedure.

Real world Training. As actions, we employ pick and place operations realized by the following sequence: moving the robot, through a motion planner, to the pick location, closing the gripper, moving to the place location, and opening the gripper. To generate an action pair, the robot performs a random action – moves a box to an adjacent tile. To create similar states, it swaps two boxes. The swapping is simply three consecutive pick and place operations. Before executing each action the robot needs to check that the preconditions of that action are true, e.g., pick location is occupied and place location is empty. This can be verified by moving and closing the gripper to the pick and place locations. If the gripper fully closes (sensed through the gripper encoder), the location is empty, otherwise a box is present and can be picked. A similar verification could be achieved with a depth camera. This formulation is consistent with the high-

Models	Dataset $\mathcal{BM}_t$							
	$\lvert\mathcal{V}\rvert$	h_c	c_c	s_c	$\lvert\mathcal{E}\rvert$	c_e	Paths scores	
							% all	*% any*
-	1016	0.92	0.92	0.79	583	0.78	0.0	0.0
PCA	496	0.75	0.78	0.52	452	0.52	0.0	0.0
AE	233	0.49	0.57	0.29	234	0.27	0.0	0.0
β-VAE	539	0.85	0.85	0.51	**422**	0.62	0.0	0.0
PC-AE	246	0.54	0.6	0.3	258	0.28	0.0	0.0
PC-VAE	570	1.0	**1.0**	0.58	488	**1.0**	29.9	29.9
PC-Sia.	389	0.99	**1.0**	0.52	458	0.97	47.37	57.3
CE-Sia.	**150**	**1.0**	**1.0**	**0.67**	325	**1.0**	**98.3**	**98.3**
	Dataset $\overline{\mathcal{BM}}_t$							
PC-AE	218	0.99	**1.0**	0.71	375	0.98	72.12	82.5
PC-VAE	395	**1.0**	**1.0**	0.56	**461**	**1.0**	89.7	89.7
PC-Sia.	**133**	**1.0**	**1.0**	**0.9**	314	**1.0**	**97.7**	**98.2**

Models	Dataset $\mathcal{BM}_{st}$							
	$\lvert\mathcal{V}\rvert$	h_c	c_c	s_c	$\lvert\mathcal{E}\rvert$	c_e	Paths scores	
							% all	*% any*
-	710	0.83	0.83	0.5	496	0.58	0.0	0.0
PCA	400	0.59	0.62	0.46	**453**	0.25	0.0	0.0
AE	554	0.87	0.88	0.56	454	0.69	0.0	0.0
β-VAE	407	0.72	0.74	0.44	361	0.38	0.0	0.0
PC-AE	318	0.62	0.91	**0.61**	325	0.47	0.0	0.0
PC-VAE	381	0.84	0.85	0.42	295	0.65	0.1	0.1
PC-Sia.	289	0.96	0.96	0.4	312	0.92	26.34	27.5
CE-Sia.	**232**	**0.99**	**0.99**	0.41	354	**0.99**	**78.39**	**78.7**
	Dataset $\overline{\mathcal{BM}}_{st}$							
PC-AE	158	0.93	0.98	**0.5**	**310**	0.79	26.01	36.4
PC-VAE	164	0.89	0.94	0.36	198	0.77	7.39	9.2
PC-Sia.	**136**	**0.99**	**0.99**	0.45	282	**0.98**	**69.37**	**72.4**

TABLE III: Evaluation results for the latent mapping models and raw observations on $\mathcal{BM}_t$ and $\mathcal{BM}_{st}$ (top) and their augmented versions (bottom) $\overline{\mathcal{BM}}_t$ and $\overline{\mathcal{BM}}_{st}$ for the box manipulation task. Best results in bold.

level actions in [1]. Using this procedure 2800 training data samples with 1330 action pairs were collected in a self-supervised manner by randomly performing actions. Note that no access to the underlying state nor human labeling is required to generate this dataset. See the supplementary video for more details.

Shelf Arrangement: As depicted in Fig. 4b), the scene of the shelf arrangement task is composed of two shelving units with a total of four shelves, and a table where four objects can be placed. Four *task-relevant* objects – a bowl, a pot, a mug and a plate (shown in the figure) – are present in the scene. This task has 70 possible states and 320 possible transitions. In addition, *distractor* objects (bottom right part of Fig. 4) can be present on the shelves and change their position. Two datasets are thus defined: *i)* $\mathcal{SA}_{0d}$, that contains the four relevant objects and zero distractor objects (2500 tuples with 1240 action pairs), *ii)* $\mathcal{SA}_{5d}$, that contains all five distractor objects with each distractor having a probability of 0.8 to appear on the shelf (2500 tuples with 1277 action pairs).

We trained each of the seven models in Sec. IV (PCA, AE, β-VAE, PC-AE, PC-VAE, PC-Siamese, and CE-Siamese) with the datasets of the above defined tasks ($\mathcal{BM}_s$, $\mathcal{BM}_t$, $\mathcal{BM}_{st}$ for box manipulation; $\mathcal{SA}_{0d}$, $\mathcal{SA}_{5d}$ for shelf arrangement) as well as their augmented versions ($\overline{\mathcal{BM}}_s$, $\overline{\mathcal{BM}}_t$, $\overline{\mathcal{BM}}_{st}$, $\overline{\mathcal{SA}}_{0d}$, $\overline{\mathcal{SA}}_{5d}$), with $n = 1$ in Sec. III. The evaluation was performed on respective holdout datasets composed of 334 and 2500 novel tuples, respectively. Further details on the architectures, hyperparameters, and additional plots can be found on the website[1] and the code[2].

VIII. Results and Discussion

Two main questions are discussed in detail:

1) Do contrastive-based losses outperform reconstruction-based losses when task-irrelevant factors of variations are present in the observations?
2) Can simple data augmentation as described in Sec. III boost the representation performance?

Influence of Contrastive Loss: To answer question 1, we carry out a quantitative and a qualitative analysis on the box manipulation task. The former is summarized in Table III (top part). We observe that models PC-VAE, PC-Siamese and CE-Siamese, employing a contrastive loss, manage to achieve almost perfect performance in terms of homogeneity (h_c), completeness (c_c) and edge score (c_e) with top view dataset $\mathcal{BM}_t$, enabling planning in their latent spaces. In particular, best planning performance (98.3% for $\%\,any$) is achieved by the pure contrastive model CE-Siamese, followed by PC-Siamese (57.3% for $\%\,any$) and PC-VAE (29.9% for $\%\,any$). In contrast, the case of no latent mapping (first row), i.e., the use of raw observations, the PCA and the reconstruction-based models achieve very low clustering and planning performance, reaching no correct paths. This also confirms the unsuitability of directly using raw high-dimensional observations for task planning with task-irrelevant factors of variations. Similarly, model PC-AE obtains poor performance, reporting $c_e = 0.28$ which leads the planning to fail due to an excessive number of erroneous edges. This suggests that the sole addition of the contrastive loss to the reconstruction one may be not sufficient to effectively structure the latent space. For the dataset $\mathcal{BM}_{st}$ (top right), having observations taken from different viewpoints, it can be noticed that the pure contrastive-based models CE-Siamese and PC-Siamese obtain the best performance in terms of clustering and planning, with CE-Siamese (78.7% for $\%\,any$) outperforming PC-Siamese (27.5% for $\%\,any$), while zero success correct paths are obtained by PC-VAE and PC-AE, mixing reconstruction and contrastive losses, as well as by PCA, AE and β-VAE. This confirms the relevance of leveraging task priors to handle task-irrelevant factors of variation, like the different viewpoints. The effectiveness of the best performing model (CE-Siamese) in regards to planning was also validated on the real robotic system shown in the supplementary video.

The above results are also reflected in the qualitative analysis in Fig. 5 (top row, left part) where the latent encodings obtained with the different models as well as raw observations ($\mathcal{O}$ column) are visualized through 2D t-SNE [38] plots. Results with $\mathcal{BM}_{st}$ are shown. We can notice that the raw observations, PCA and purely reconstruction-based models AE and β-VAE fail in structuring the representations,

[2]https://github.com/State-Representation/code

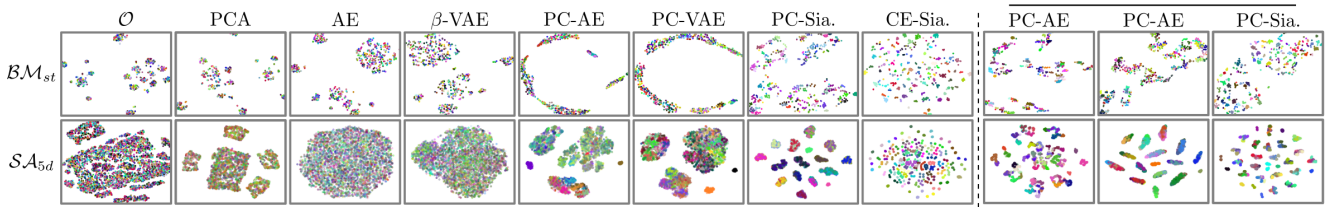

Fig. 5: Two-dimensional t-SNE plots for the box manipulation task for the mixed view (top row) and the shelf arrangement task for the distractor case (bottom row). Each color is associated with a possible underlying state. On the right we display the plots for the augmented contrastive models. Full results are accessible on the website[1].

Models	Dataset $\mathcal{SA}_{0d}$							
	$\lvert\mathcal{V}\rvert$	h_c	c_c	s_c	$\lvert\mathcal{E}\rvert$	c_e	Paths scores	
							% all	*% any*
PC-AE	5	0.28	**1.00**	0.93	4	0.75	0.00	0.00
PC-VAE	5	0.28	**1.00**	0.78	4	0.75	0.00	0.00
PC-Sia.	**16**	0.64	**1.00**	**0.96**	**32**	0.59	1.01	1.8
CE-Sia.	296	**1.00**	**1.00**	0.54	842	**1.00**	**95.90**	**95.90**
	Dataset $\overline{\mathcal{SA}}_{0d}$							
PC-AE	97	0.87	0.95	0.51	**368**	0.77	20.30	35.90
PC-VAE	**64**	0.90	0.99	0.32	235	0.77	33.13	55.40
PC-Sia.	225	**1.00**	**1.00**	**0.74**	772	**1.00**	**100.0**	**100.0**

Models	Dataset $\mathcal{SA}_{5d}$							
	$\lvert\mathcal{V}\rvert$	h_c	c_c	s_c	$\lvert\mathcal{E}\rvert$	c_e	Paths scores	
							% all	*% any*
PC-AE	5	0.28	**1.00**	0.9	4	0.75	0.40	0.40
PC-VAE	5	0.28	**1.00**	0.79	4	0.75	0.40	0.40
PC-Sia.	**16**	0.64	**1.00**	**0.98**	**32**	0.69	2.42	4.10
CE-Sia.	286	**1.00**	**1.00**	0.46	841	**0.99**	**95.12**	**95.30**
	Dataset $\overline{\mathcal{SA}}_{5d}$							
PC-AE	18	0.49	0.98	0.01	34	0.44	0.24	0.40
PC-VAE	16	0.64	**1.00**	0.52	32	0.69	2.42	4.10
PC-Sia.	**30**	**0.78**	**1.00**	**0.61**	**87**	**0.74**	**6.66**	**13.10**

TABLE IV: Evaluation results for the contrastive-based latent mapping models on $\mathcal{SA}_{0d}$ (top left) and $\mathcal{SA}_{5d}$ (top right) and their augmented versions $\overline{\mathcal{SA}}_{0d}$ (bottom left) and $\overline{\mathcal{SA}}_{5d}$ (bottom right) for the shelf arrangement task. Best results in bold.

forming spurious clusters in which different states are mixed up. Non-homogeneous clusters are also obtained by PC-AE and PC-VAE, while a significant improvement of the latent space structure is recorded by the purely contrastive loss based Siamese networks (PC-Siamese and CE-Siamese), leading to visually distinct clusters.

In summary, we observe that the contrastive-based models (PC-Siamese, CE-Siamese) outperform the other ones by a significant margin. Notably, the architectures of the Siamese networks are much shallower[2] *than the AE and VAE ones, leading to considerably faster training processes* ($<$ 3.5 minutes vs $\approx$ 2.5 hours on a GeForce GTX 1080 Ti).

Influence of Data Augmentation: To evaluate the influence of the data augmentation in Sec. III, we first analyze the representation performance on the shelf arrangement task when it is applied and when it is not. For the sake of space, we focus only on the four contrastive-based models since the unsuitability of raw observations, PCA, AE and β-VAE has been shown above. Table IV reports the obtained evaluation metrics. When no augmentation is applied (top part), all the models, except for CE-Siamese, show very low performance for both clustering and planning on both datasets, creating a small number of clusters ($\ll 70$) that are erroneously connected. In contrast, CE-Siamese generates a large amount of pure clusters ($\simeq 300$ clusters with $h_e \simeq 1$ for both datasets) which are almost perfectly connected ($c_e \simeq 1$), leading to high path metrics (*% any* $\simeq 95\%$ for both datasets). When the augmentation is used, the performance of all models improves for the no distractors dataset (bottom left), leading the PC-Siamese to reach perfect path metrics (100% for *% any*) and PC-AE and PC-VAE to reach $\simeq 36\%$ and $\simeq 55\%$ for *% any*. This confirms the beneficial effect of the considered augmentation which, however, is not equivalently effective when distractor objects are present in the scene (bottom right). More specifically, only PC-Siamese is positively influenced by the augmentation with $\overline{\mathcal{SA}}_{5d}$, reaching path metrics *% any* $\simeq 13\%$ (from $\simeq 4\%$). This suggests that a higher number of dissimilar pairs should be synthetically generated for this case study, i.e., $n \gg 1$. Note the augmented datasets are only used for the latent mapping but not for the LSR building to avoid building wrong edges. Moreover, the CE-Siamese is not evaluated with the augmentation technique since it does not use action pairs. Similar observations also hold for the box manipulation dataset in Table III, where we can notice that, when the augmentation is used (bottom part), PC-Siamese manages to achieve almost perfect performance on the top view dataset $\mathcal{BM}_t$, with $|\mathcal{V}| = 133$ clusters and path performance 98.2% for $\%\,any$, as well as good performance on the mixed view dataset $\mathcal{BM}_{st}$, with path performance 78.8% for $\%\,any$. General improvements are also recorded for PC-AE and PC-VAE which, however, underperform the purely contrastive-based models.

Fig. 5 reports the t-SNE plots for the shelf stacking task (bottom row) with five distractors ($\mathcal{SA}_{5d}$) obtained with (on the right) and without (on the left) augmentation. In this task the optimal number of clusters is 70. It is evident from the t-SNE visualizations that, in the absence of augmentation, only the CE-Siamese model can structure the encodings such that clusters of different states are not overlapping. This is due to the training procedure of CE-Siamese [26], which only relies on similar pairs and synthetically builds a large number of dissimilar pairs [26]. In contrast, better separation of the states is observed with data augmentation. Notably, in $\overline{\mathcal{SA}}_{5d}$, PC-Siamese, which solely relies on the contrastive loss, achieves a better clustering than PC-AE and PC-VAE, which also exploit reconstruction loss. Similar considerations also hold for the box manipulation task (top row of Fig. 5).

In summary, we observe that a simple data augmentation boosts the performance of the contrastive-based models.

IX. Conclusion

In this work, we investigated the effect of different loss functions for retrieving the underlying states of a system from visual observations applied to task planning. We showed that purely reconstruction-based models are prone to fail when task-irrelevant factors of variation are present in the observations. In contrast, the exploitation of task priors in contrastive-based losses as well as of an easy data augmentation technique resulted in a significant representation improvement. We analyzed two robotics tasks with different task-irrelevant factors of variation: *i)* box manipulation, on a real robotic system with different viewpoints and occlusions, and *ii)* shelf arrangement, with distractor objects that are irrelevant for the task itself. We thus believe that contrastive-based losses as well as simple data augmentations go a long way toward obtaining meaningful representations that can be used for a wide variety of robotics tasks and provide a promising direction for the research community.

References

[1] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *J. Artificial Intelligence Res.*, vol. 61, pp. 215–289, 2018.

[2] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, "Decoupling representation learning from reinforcement learning," *arXiv preprint arXiv:2009.08319*, 2020.

[3] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach." in *Robotics: Science and Syst.*, vol. 12, 2016, p. 00052.

[4] V. Thomas, E. Bengio, W. Fedus, J. Pondard, P. Beaudoin, H. Larochelle, J. Pineau, D. Precup, and Y. Bengio, "Disentangling the independently controllable factors of variation by interacting with the world," *Learning Disentangling Representations Wksp. at NeurIPS*, 2017.

[5] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robot. Autom. Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[6] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *Adv. Neural Inf. Process. Syst.*, vol. 28, pp. 2746–2754, 2015.

[7] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Int. Conf. on Mach. Learn.*, 2019, pp. 2555–2565.

[8] K. Pertsch, O. Rybkin, F. Ebert, S. Zhou, D. Jayaraman, C. Finn, and S. Levine, "Long-horizon visual planning with goal-conditioned hierarchical predictors," *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020.

[9] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *Int. Conf. on Learn. Representations*, 2019.

[10] S. Nair, S. Savarese, and C. Finn, "Goal-aware prediction: Learning to model what matters," in *Int. Conf. on Mach. Learn.*, 2020, pp. 7207–7219.

[11] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *IEEE Int. Conf. Robot. Autom.*, 2016, pp. 512–519.

[12] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. K. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Visuospatial foresight for multi-step, multi-task fabric manipulation," *Robotics: Science and Syst.*, 2020.

[13] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2016, pp. 3928–3934.

[14] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, p. 193907–193934, 2020.

[15] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *Int. Conf. on Mach. Learn.*, 2020, pp. 5639–5650.

[16] T. Kipf, E. van der Pol, and M. Welling, "Contrastive learning of structured world models," in *Int. Conf. on Learn. Representations*, 2019.

[17] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1134–1141.

[18] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *Int. Conf. on Learn. Representations*, 2020.

[19] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, no. 3, pp. 407–428, 2015.

[20] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[21] R. Legenstein, N. Wilbert, and L. Wiskott, "Reinforcement learning on slow features of high-dimensional input streams," *PLoS computational biology*, vol. 6, no. 8, p. e1000894, 2010.

[22] S. Höfer, M. Hild, and M. Kubisch, "Using slow feature analysis to extract behavioural manifolds related to humanoid robot postures," in *Int. Conf. on Epigenetic Robot.*, 2010, pp. 43–50.

[23] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic, "Latent space roadmap for visual action planning of deformable and rigid object manipulation," *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2020.

[24] N. Jetchev, T. Lang, and M. Toussaint, "Learning grounded relational symbols from continuous data for abstract reasoning," 2013.

[25] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *IEEE Int. Conf. Robot. Autom.*, 2017, pp. 2786–2793.

[26] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, 2020, pp. 1597–1607.

[27] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *J. Educational Psychology*, vol. 24, no. 6, p. 417, 1933.

[28] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[29] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[30] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun, "Unsupervised feature learning from temporal data," *arXiv preprint arXiv:1504.02518*, 2015.

[31] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, 2006, pp. 1735–1742.

[32] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Adv. Neural Inf. Process. Syst.*, 2016, pp. 1857–1865.

[33] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *J. Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[34] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Joint Conf. Empirical Methods in Natural Language Process. and Computational Natural Language Learn.*, 2007, pp. 410–420.

[35] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *J. Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[36] J. K. Haas, "A history of the unity game engine," 2014.

[37] C. Chamzas, M. Lippi, M. C. Welle, A. Varava, A. Marino, L. E. Kavraki, and D. Kragic, "State representations in robotics: Identifying relevant factors of variation using weak supervision," in *Robot Learn. Wksp. at NeurIPS*, 2020.

[38] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.