



Doctoral Thesis in Computer Science

Learning Structured Representations for Rigid and Deformable Object Manipulation

MICHAEL C. WELLE

Learning Structured Representations for Rigid and Deformable Object Manipulation

MICHAEL C. WELLE

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Thursday the 9th December 2021, at 3:00 p.m. in , Ångdomen, Osquars backe 31, KTH Campus, Stockholm

Doctoral Thesis in Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden 2021

© Michael C. Welle

ISBN 978-91-8040-050-3
TRITA-EECS-AVL-2021:72

Printed by: Universitetsservice US-AB, Sweden 2021

Für Helga und Christoph

Abstract

The performance of learning based algorithms largely depends on the given representation of data. Therefore the questions arise, *i*) how to obtain useful representations, *ii*) how to evaluate representations, and *iii*) how to leverage these representations in a real-world robotic setting. In this thesis, we aim to answer all three of these questions in order to learn structured representations for rigid and deformable object manipulation. We firstly take a look into how to learn structured representation and show that imposing structure, informed from task priors, into the representation space is beneficial for certain robotic tasks. Furthermore we discuss and present suitable evaluation practices for structured representations as well as a benchmark for bimanual cloth manipulation. Finally, we introduce the Latent Space Roadmap (LSR) framework for visual action planning, where raw observations are mapped into a lower-dimensional latent space. Those are connected via the LSR, and visual action plans are generated that are able to perform a wide range of tasks. The framework is validated on a simulated rigid box stacking task, a simulated hybrid rope-box manipulation task, and a T-shirt folding task performed on a real robotic system.

Sammanfattning

Prestandan av inlärningbaserade algoritmer beror på stor del av hur datan representeras. Av denna anledning ställs följande frågor: *(i)* hur vi tar fram användarbara representationer, *(ii)* hur utvärderar vi dem samt *(iii)* hur kan vi använda dem i riktiga robotikscenarier. I den här avhandlingen försöker vi att svara på dessa frågor för att hitta inlärda, strukturerade, representationer för manipulation av rigida och icke-rigida objekt. Först behandlar vi hur man kan lära in en strukturerad representation och visar att inkorporering av struktur, genom användandet av statistiska priors, är fördelaktigt inom vissa robotikuppgifter. Vidare så diskuterar vi passande tillvägagångssätt för att utvärdera strukturerade representationer, samt presenterar ett standardiserat test för tygmanipulering för robotar med två armar. Till sist så introducerar vi ramverket Latent Space Roadmap (LSR) för visuell beslutsplanering, där råa observationer mappas till en lågdimensionell latent rymd. Dessa punkter kopplas samman med hjälp av LSR, och visuella beslutsplaner genereras för en simulerad uppgift för att placera objekt i staplar, för manipulation av ett rep, samt för att vika T-shirts på ett riktigt robotiksystem.

Acknowledgements

These past four years of my Ph.D. have been a greatly positive experience in general ($\approx 77\%$). This would not have been the case would it not have been for a large number of people that collaborated with me, shared my interests and personal time with me, as well as those that provided useful supervision for me. I want to use these next lines to thank and acknowledge all those that contributed to the time being the positive experience that it was and also thank the people that helped me on my way in getting here in the first place.

First of course I want to thank my main supervisor, Dani, for giving me the opportunity to not only pursue a Ph.D. in her group but also allow me the freedom to explore the areas I found most interesting. Thank you for providing counsel not only regarding the direction of my research but also considering collaborations and long-term career goals.

I also want to thank the co-supervisors that contributed positively to my Ph.D: Firstly, and most importantly, Anastasia, you are probably the single most positive influence of the last 4 years of my life. Thank you for being my co-supervisor and friend. You were there to help me through tough times as well as celebrate good times with me. Having you over at my place for dinner and whisky (among other things) constituted some of the best times during my Ph.D. Gaining you as a friend enriched my life and I'm looking forward to it continuing to do so. Thank you for being who you are.

Secondly, I also want to thank my other co-supervisor Hang, simply discussing a problem with you often helps coming closer to the solution. Your vast knowledge about any research direction I'm interested in was very helpful to make the contributions that I did. Also thanks for providing additional insights into the realm of Portuguese Tinto. A short thanks is also extended to Robert, your supervision at the beginning of my Ph.D. was short but appreciated nonetheless.

During my time as a Ph.D. student, I had the pleasure to collaborate with 25 unique collaborators from four different universities, where 96% contributed in a positive manner. For the sake of space, I will not address each of the positive collaborators individually but be reassured that your contribution made the final paper better than it would have been without you and I thank you for your contributions. This been said, some collaborators had a larger impact than others and I want to specifically thank the Foldinggirls (FG's)! I want to thank Martina, for being an amazing person to work with. The projects with you are always a joy and together we are productive and in sync as I have experienced only very very seldom in my life so far. I consider myself exceptionally lucky that you selected KTH as the destination for your research visit, and that you choose to continue the collaborations even from Italy. I not only gained a superb collaborator over the time of our projects but also a dear friend that I value and hope to get to make more pancakes for. The other FG Petra (equally attractive), deserves similar thanks, as we are able to complement each other's areas of expertise as well as our

personalities. You also have a (positive) influence on me in regards to sports and food. I'm very thankful you continue to light up the world and are always available for a quick discussion and ready to bounce ideas off each other or for a afterwork sushi and/or beer. Also, I'm glad that there is finally another person that can appreciate fine single malt Whisky!

I also want to thank Judith, while we only collaborated to a small amount having you as a friend has enriched my life and I hope that I will develop the bread-making skills worthy of your approval. A special thanks goes also out to Constantinos, thanks for being easy and fun to work with and I hope we see each other in person eventually. I was lucky enough to spend my Ph.D. in the RPL lab, which exhibits a surprising (for academia) nice and inviting atmosphere. I want to especially thank Joao and Vlad for participating in many fun tastings as well as having great taste in Movies. Joao - I will never forget the beautiful sunsets we saw at 101. Vlad - thanks for providing a taste profile regarding liquor that can be used as a prediction model with high reliability. I also want to thank my RPL roommates over the years, Erik and Akshaya for welcoming me into the office and answering my many questions in a generally helpful way. I also want to thank Jonny for moving in eventually and providing a fun atmosphere as well as reminding me that my skills in Smash are still very lacking. The spirit of how great RPL can be is perfectly captured by Jana which I also want to extend my thanks to for helping despite being super busy and being a generally pleasant person to be around. Ludvig also deserves a shoutout for not only being fun to be around during the Ph.D. but also already during the masters. Similarly Isac deserves thanks for helping me with MPC as well as his ongoing quest to teach me puns in the swedish language. Yosuha and Josuha also deserve thanks, one for being the happiest person I ever encountered the other for showing that you can always have more experiments in your papers. I also want to thank Vlad (different Vlad), Özer, Silvia, and Alex for playing badminton during the beginning of my Ph.D. For more sports, I also want to thank the Italian gang: Alberta, Alfredo, and Marco for playing squash as well as being able to now start working on interesting projects together. I hope your Ph.D. will bring you at least as much joy as it did for me. And I thank everyone else that helps make RPL a good place to do a Ph.D at.

Outside of RPL, I want to extend my first thanks to Nadine - thank you for coming to Sweden with me and taking on the Master, and thank you for convincing me I am capable of doing the Ph.D. as well. I would not have succeeded if it would not have been for you. I am the person I am today thanks to you and I will always remember our time together fondly. I also want to thank Rares and old Hang for giving me insights into the Ph.D. life to make my decision more informed. Furthermore, I want to thank Daniel and his wife Anna, for welcoming me to Sweden and teaching me about the staples of Swedish cosine such as tacos and lemon pie. I also want to thank my friends Jakob and Emma - thank you guys especially for being there during study times in the Masters as well as improving many evenings with your presents.

I also want to thank my German friends, first my clan 176 mate TT99 for

bridging the gap to Sweden using games such as EEZDE and Breakpoint. A special thanks goes also out to the member of the Eiersalat group, Medhi, Treiber, Key, Gutman, and Thomas for some nice gatherings. The sentiment is extended to Maike, Miri, and Bierle for always being fun to be around. Finally the Bredouillatoren group: Simon Haas, schöne Thomas, Michix2, Stefan, Flo, and Methler as well as Reinsen. I hope to have a beer again with all of you soon.

Zum schluss möchte ich auch noch meiner fammily danken. Meiner Mutter Helga und meinem Vater Christoph ohne die buchstäblich nichts von all dem möglich gewesen wäre. Meinen Schwestern Katherina and Rebecca dafür das ihr die besten schwestern seit wo man sich denken kann. Und meinen Großeltern, Oma und Opa Resel für stetiges freudiges beisamsein und guten Wein, und Mutsch für das austrahlen von lebensfreüde selbst sowie köstliche Maultaschen.

Michael C. Welle

Contents

I Overview	1
1 Introduction	3
1.1 Scope of the Thesis	3
1.2 List of Papers	4
2 Learning Structured Representations	9
2.1 Overview of Existing Approaches	9
2.2 Motivation and Problem Definition: Structured Representations for Object Manipulation	12
2.3 Feature Extraction and Structuring Representations	16
2.4 Evaluation of Frameworks and Structured Representations	18
2.5 Latent Space Roadmap - An Example of a Combined Framework	20
3 An Overview of Publications	25
4 Conclusion and Future Work	31
4.1 Cage-Flow - A Flow-based Cage Proposal Model	31
4.2 Hierarchical-LSR - A LSR framework for Continuous Problems	32
Bibliography	35
II Included Publications	43
A Partial Caging: A Clearance-Based Definition, Datasets, and Deep Learning	A1
B Enabling Visual Action Planning for Object Manipulation through Latent Space Roadmap	B1
C Benchmarking Bimanual Cloth Manipulation	C1
D Textile Taxonomy and Classification Using Pulling and Twisting	D1
E Comparing Reconstruction- and Contrastive-based Models for Visual Task Planning	E1

Part I

Overview

Chapter 1

Introduction

1.1 Scope of the Thesis

Object manipulation has been a central focus of a large part of robotic research [1]. Manipulation of rigid objects in an industrial environment is addressed by heavily controlling the environment and employing manipulators with high accuracy and repeatability. Performing the same complex manipulation in semi-structured environments such as shared workspaces between robots and humans, or completely unstructured and unpredictable environments as found in people’s homes remains an open problem. Moreover, rigid object manipulation has received the bulk of the research attention in earlier works, while the handling of deformable objects only recently enjoyed more and more considerations. One reason is the difficulty in estimating and controlling the state of deformable objects. The degrees of freedom of a deformable object are infinite in many cases and an analytical description is often unfeasible. This facts alone makes transferring classical approaches developed for rigid object manipulation that rely on exact state estimation and descriptions unfeasible. With the onset of deep learning, a powerful new tool was added to the robotics researcher’s toolbox. Learning-based methods are a promising direction to tackle the challenges of very high dimensional state-spaces and how to describe the object state, assisting in the solution to challenging manipulation tasks such as handling deformable objects. While these methods have their own shortcomings, the need for an extreme amount of data in an end-to-end setting or a general lack of interpretability, the ability to automatically extract beneficial features from the provided data and encode them into internal representations is a particularly intriguing trait.

Specifically, the field of representation learning aims to *represent* given input data in a favorable format that can be efficiently exploited for further downstream tasks. Downstream tasks in the computer vision community are defined as “... computer vision applications that are used to evaluate the quality of features learned by self-supervised learning. These applications can greatly benefit from the pretrained models when training data are scarce. [2]”. In robotics, a downstream task can be

any task that uses the given representations to achieve a goal such as reaching, pushing, stacking, folding, etc. In general an example of a simple representation learning scenario involves a supervised feedforward neural network, weights in the last layer can be thought of as representations to perform the downstream classification task. When updating these weights during the learning process, they are shaped into a more favorable representation. Therefore a simple way of obtaining a potentially useful representation is to take the internal representation of a certain layer of a supervised trained model. The question of what layer to take as representations from a pre-trained model highlights an important aspect of representation learning, if an early layer is used the representations are still high dimensional but are more likely to contain all the relevant information. The deeper the layer the more compact the representation becomes but the higher the risk that crucial information that is necessary for the success of the downstream task is missing. Therefore this approach can only succeed given that the structure of these internal representations manage to strike the trade of between complexity and richness of information preserved. As when using a too complex representation that contains all necessary (and unnecessary) information the downstream learner has to be very expressive and can be difficult to train.

Furthermore when considering robotic tasks such as folding one can also impose explicit constraints on these intermediate representations in a more direct manner thus imposing structure into the representation. The information needed to impose the wanted constraints can be informed by task priors instead of explicit class labels like in a supervised learning setting. An aspiring goal for representation learning is therefore to produce representations that can be leveraged by weak networks or methods that rely on more classical approaches to address a given task. This requirement often requires imposing an additional degree of structure into the learned representation, as the input needs to be fitting for the employed downstream method.

In this thesis, we address the following two specific questions:

- How to learn structured representations for rigid and deformable object manipulation?
- How to leverage the learned structured representations for improving the performance of challenging robotic tasks such as folding in a real-world setting?

1.2 List of Papers

The following is a list of all papers this thesis is based upon. Three papers are published papers (Chapters A, C, D) and two papers currently under review (Chapters B, E) where paper B is an extension of the published paper (X-3). Papers not included are denoted X-1 to X-5. We indicate the respective robotics venue as well as specify the contribution.

Paper A: Partial Caging: A Clearance-Based Definition, Datasets, and Deep Learning (Extension of paper X-2)

Michael C. Welle, Anastasiia Varava, Jeffrey Mahler, Ken Goldberg, Danica Kragic and Florian T. Pokorny

Published in Autonomous Robots, 1-18, 2021

Contributions by the author: Developed a partial caging metric, designed and implemented neural network approximation of partial caging method, developed partial caging acquisition pipeline, performed experimental evaluation and ablation study, wrote the majority of the paper.

Paper B: Enabling Visual Action Planning for Object Manipulation through Latent Space Roadmap (Extension of paper X-3)

Martina Lippi*, Petra Poklukar*, **Michael C. Welle***, Anastasia Varava, Hang Yin, Alessandro Marino, and Danica Kragic

Under review in IEEE Transactions on Robotics, 2021

Contributions by the author: formulated and proposed the problem, major contribution to algorithmic design and implementation, performed major parts of experimental evaluation and ablation study, wrote major parts of the paper.

Paper C: Benchmarking Bimanual Cloth Manipulation

Irene Garcia-Camacho*, Martina Lippi*, **Michael C. Welle**, Hang Yin, Rika Antonova, Anastasia Varava, Julia Borrás, Carme Torras, Alessandro Marino, Guillem Alenyà and Danica Kragic

Published in IEEE Robotics and Automation Letters, 5, 1111-1118, 2020

Contributions by the author: formulated and proposed benchmark tasks and evaluation procedure, performed major parts of experimental evaluation, wrote parts of the paper.

Paper D: Textile Taxonomy and Classification Using Pulling and Twisting

Alberta Longhini, **Michael C. Welle**, Ioanna Mitsioni and Danica Kragic

Published in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021

Contributions by the author: formulated and proposed the problem, designed parts of the experiment, setup the robotic platform, wrote minor parts of the paper.

* Authors contributed equally, listed in alphabetical order.

Paper E: Comparing Reconstruction- and Contrastive-based Models for Visual Task Planning

Constantinos Chamzas*, Martina Lippi*, **Michael C. Welle***, Anastasia Varava, Lydia E. Kavraki, and Danica Kragic

Under review in IEEE Robotics and Automation Letters, 2021

Contributions by the author: formulated and proposed the problem, major contribution to algorithmic implementation, performed major parts of experimental evaluation and ablation study, wrote major parts of the paper.

The following published papers are not included in the thesis:

Paper X-1: From visual understanding to complex object manipulation

Judith Bütepage, Silvia Cruciani, Mia Kokic, and **Michael C. Welle**, and Danica Kragic

Published in Annual Review of Control, Robotics, and Autonomous Systems, 2, 161 - 179, 2019

Contributions by the author: wrote minor parts of the paper about robotic tool use, proofread the paper and contributed to the scientific discussion.

Paper X-2: Partial Caging: A Clearance-Based Definition and Deep Learning

Anastasiia Varava*, **Michael C. Welle***, Jeffrey Mahler, Ken Goldberg, Danica Kragic and Florian T. Pokorny

Published in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1533-1540, 2019

Contributions by the author: Developed a partial caging metric, designed and implemented neural network approximation of partial caging method, performed experimental evaluation, wrote major of the paper.

Paper X-3: Latent Space Roadmap for Visual Action Planning of Deformable and Rigid Object Manipulation

Martina Lippi*, Petra Poklukar*, **Michael C. Welle***, Anastasia Varava, Hang Yin, Alessandro Marino, and Danica Kragic

Published in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 5619-5626, 2020

Contributions by the author: formulated and proposed the problem, major contribution to algorithmic design and implementation, performed major parts of experimental evaluation and ablation study, wrote major parts of the paper.

* Authors contributed equally, listed in alphabetical order.

Paper X-4: Fashion Landmark Detection and Category Classification for Robotics

Thomas Ziegler, Judith Bütepage, **Michael C. Welle**, Anastasiia Varava, Tonci Novkovic, and Danica Kragic

Published in 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 81-88, 2020

Contributions by the author: Designed major parts of experiments, implemented and performed minor experiments, wrote minor parts of the paper.

Paper X-5: Learning Task Constraints in Visual-Action Planning from Demonstrations

Francesco Esposito, Christian Pek, **Michael C. Welle**, and Danica Kragic

Published in 2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN), 131-138, 2021

Contributions by the author: formulated and proposed the problem, designed major parts of the experiments, wrote minor parts of the paper.

Chapter 2

Learning Structured Representations

This chapter first provides a overview of existing approaches concerning the learning of representations. We follow by providing the problem definition as well as the motivation for employing addition structure. Next the three question posed at the beginning, how to obtain, how to evaluate and how to leverage structured representation in a real world setting are addressed and related to the included publications.

2.1 Overview of Existing Approaches

A good representation to facilitate the manipulation of objects is one that makes it easier to perform state estimation, dynamic prediction, and planning. In [3] the authors perform an exhaustive review focused on deformable object manipulation. The authors split existing methods presented into two distinct groups, model-based and data-driven methods. Model-based approaches encompass works that base their models of deformable objects on physics-based approaches such as mass-spring systems, point-based dynamics, and continuum mechanics. In contrast the data-driven approaches infer the required properties directly from the data. These different approaches often have an inherent trade-off between model accuracy, robustness, and computational costs. A combination of physics-based model approaches has been integrated into simulation environments [4], [5], [6], [7], [8], with more and more focusing specifically on deformable objects recently. While these simulators have achieved significant progress in recent years, the formentioned trade-offs have to be considered when coupling them with deformable manipulation methods. Many approaches are opting therefore to learn their own, often task-specific representation.

The data-driven approaches can be divided into End-to-End and latent planning methods. In End-to-End methods, there is no explicit separation between the representation produced and the downstream task, while in latent planning methods

the representations are constructed first, and planning is done leveraging these representations. We will also discuss a subset of the latent planning methods called visual action planning, since they are at the focus of this thesis.

End-to-End: The End-to-End setting was to a large extent popularised with the introduction of Deep Q-Networks (DQN) [9], [10]. Which achieved average human-level performance on a large number of Atari games taking the raw observations as input and outputting directly the control actions required to play the games. The success of DQN established deep reinforcement learning as a separate research direction and spawned a number of now widely used methods such as for example Deep Deterministic Policy Gradients (DDPG) [11], Hindsight Experience Replay (HER) [12], and Proximal Policy Optimization (PPO) [13].

One of the earliest applications of deep reinforcement learning onto the field of object manipulation was presented in [14]. The authors framed the problem of manipulation skills as a policy search problem where the policy is producing a probability distribution over possible actions given the current state. Leveraging this formulation, the authors were able to demonstrate that their method can learn controllers that are robust to small perturbations for contact-rich tasks such as stacking Lego blocks or screwing caps on bottles. In [15], the authors take the End-to-End deep reinforcement learning approach towards the manipulation of deformable objects. DDPG was leveraged in order to facilitate policy learning in simulation with domain randomization, which was then applied to a real system. The authors considered manipulation tasks using a cloth towel; folding the towel as well as draping it over a hanger. The policy learned in simulation is then to be applied on a real-world system (Sim-to-Real). While Sim-to-Real is demonstrated to be possible using a simulation environment and domain randomization, seeding the training with demonstration is necessary for the method to succeed. The quality of the simulated data is a large potential impediment of data-hungry End-to-End approaches as getting accurate simulation data of more complex clothing items is still an open problem.

Latent planning: In latent planning the goal is to create a representation that is not only useful for neural networks but can also be leveraged by established planning and control methods, thereby aiming to use the best of two worlds. Deep learning methods are employed to deal with the sparse high dimensional observations (often in form of images) that the system receives, while tried and proven concepts from the planning and control field are subsequently used directly on the latent representations. In [16] one of the landmark frameworks is introduced: Embed to Control (E2C), where a Variational Autoencoder (VAE) [17] is used to learn so called image trajectories from a latent space where the dynamics are constrained to be locally linear using an optimal control formulation. The authors showcased their method on a simple planar system, swing-up of an inverted pendulum, balancing a cart-pole, and a two-jointed robotic arm. They furthermore tested enforcing temporal slowness during the learning process, by augmenting the standard VAE loss function with an additional Kullback-Leibler (KL)-divergence [18] term that enforces

temporally close images to also be encoded close in the latent space. While the authors showed a slightly better coherence of similar states, they remarked that the slowness term alone is not sufficient to impose the required structure in the latent space. The idea of learning latent dynamics from raw observation is also discussed in [19], where a Deep Planning Network (PlaNet) was proposed, with the goal of modeling the environment dynamics from experience. The method is therefore iteratively collecting data by first performing a model fitting step on the data collected so far and then using the achieved planning performance to collect additional data. The authors showed that a recurrent state-space model is most successful when splitting the state into a stochastic and deterministic part. Another core insight is to roll out multiple latent prediction steps and to incorporate the results into the loss function. To plan using the latent model PlaNet employs the cross entropy method (CEM [20], [21]) that aims to recover a probability distribution over actions given the model. Employing a sample-based motion planning method on the other hand was presented in [22]. In detail, the authors proposed to leverage an Autoencoder (AE) [23] network to construct the latent space, a dynamics network, and a collision checking network. These three components are then combined to perform sampling-based motion planning - namely rapidly-exploring random tree (RRT). The authors showcased their learned latent RRT (L2RRT) on a top-down visual planning problem, as well as on a humanoid robot as an example of a high-dimensional dynamical system. Using the latent representation directly for planning is, however, not the only way to leverage the representation’s advantages. Another way is to deploy graph structures on top of the latent space to make planning more suitable for the task at hand. For example, in [24] the idea of employing a graph directly in the latent space was considered. The authors encoded every new observation obtained in a 3D maze exploration setting. Given a trajectory, the images are encoded using an encoder network and added to a memory graph, where subsequent encodings are connected to each other. A number of shortcuts are then build to connect the encodings of nodes that are close to each other. This procedure results in a graph with a large number of nodes, where a single step can sometimes be too small of an increment to produce a meaningful action. The authors address this by employing a hyperparameter to select a sensible waypoint to traverse.

Visual Action Planning: While planning can be performed directly on latent representations, there are additional benefits if the method is also able to visualize the planned steps in the form of images. In visual action planning, the goal is to produce not only a sequence of actions but also a sequence of images showing the intermediate states on the path towards reaching goal. These images can be used to perform the control step directly, for example as in [25], where a video prediction model is fed with a large number of raw observations from self-supervised object manipulation in a real robot scenario. The model is conditioned on the corresponding actions making it possible to generate images depending on planned actions. A Model Predictive Control (MPC) approach was then implemented directly on the generated images. Interestingly the same authors observed in [26], a follow-

up work where they generated sub-goal images to guide the planning, that using the pixel distance was more beneficial than using representations. Work exploiting the images directly is also presented in [27] where a model based upon the Causal InfoGAN framework was used in order to generate visually plausible images from start to goal state using data obtained in self-supervision. The “imagined” path is then used in a visual servoing controller to move the rope into the desired configuration. The authors performed background removal and gray-scaling to make the generated images easier to achieve and be invariant to potential color changes of the rope. [28] went a step further and directly learned a state representation of the rope consisting of 64 rope segments. This was achieved by first estimating 8 straight segments using the Visual Geometry Group (VGG) network [29] followed by using three consecutive spatial transformer networks [30] to double the number of segments at every step. Self-supervised approaches are, however, not limited to 1D deformable objects like ropes. In [31] not only a rope was considered, but also a 2D square cloth. In this case, the authors were able to learn the latent dynamics for these deformable objects from simulation. Here the InfoNCE contrastive loss [32] was used, where a similar pair is defined as having a (small) action between the states, while a negative/dissimilar pair is a given observation to any other observation in the dataset. Note that this notion of similar pairs holds as long as the perpetuating action is very small. Simulated data of square cloth items was also leveraged in [33]. Here, no task demonstrations were needed and the CEM was used to select promising action sequences. The authors not only considered specific folds for the piece of cloth but also smoothing it from a crumpled starting position. While we employ similar and dissimilar pairs in our works as well, in contrast to [31] we employed higher-level action, that results in a guaranteed state change and defined these pairs therefore as dissimilar pairs. The definition of the similar pairs is in agreement with [31] by having small permutations between the observations.

2.2 Motivation and Problem Definition: Structured Representations for Object Manipulation

Before going into the details of learning structured representations that are favorable for object manipulation, we clarify certain assumptions and desired properties that the representations should exhibit.

- i)* We assume that our method has access to observations and can change the underlying states of the system by performing known actions.
- ii)* We assume that we have access to enough observations and transition between some of the observations to cover the task-relevant part of the system.

This assumptions are illustrated in Fig. 2.1, and formalized in the following:

Assumption 1. *Let the underlying system states $x \in \mathcal{X}$ of a system \mathcal{X} be hidden, let \mathcal{A} be the set of all possible actions for \mathcal{X} , and let \mathcal{Z} be the representation space of \mathcal{X} . Let \mathcal{O} be the space of all possible observations of a given system \mathcal{X} , Let there*

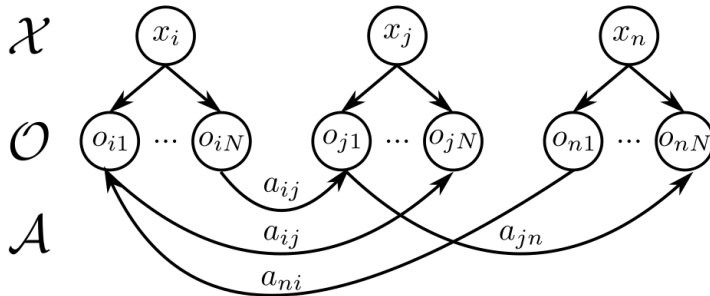


Figure 2.1: A hidden state $x \in \mathcal{X}$ in a system \mathcal{X} can emit a number of different observations $o \in \mathcal{O}$. An agent can interact with the system by performing actions $a \in \mathcal{A}$.

be a random agent $f(\cdot)$ that can interact with the system by performing a known random action $a \in \mathcal{A}$ changing the system's state.

We are interested in obtaining a mapping $\xi : \mathcal{O} \rightarrow \mathcal{Z}$. Before detailing the mapping ξ we define what kind of properties are desirable for the representation space \mathcal{Z} .

Representation properties: As [34] points out, a *good* representation is “... one that captures the posterior distribution of the underlying explanatory factors for the observed input [34].” Furthermore, a good representation should also be useful as input to a supervised predictor. [35] points to the particular case of state representation learning, which focuses on learned features of low dimensionality, evolving through time and are influenceable by the actions of an agent. Similarly the authors outline the desired state representations in [36] “...

- *Markovian, i.e. it summarizes all the necessary information to be able to choose an action within the policy, by looking only at the current state.*
- *Able to represent the true value of the current state well enough for policy improvement.*
- *Able to generalize the learned value-function to unseen states with similar futures.*
- *Low dimensional for efficient estimation. ”*

Note that the desire for structured representation is not explicitly stated.

The Need for Structured Representation

As we can see in Fig. 2.1, the states x_i, \dots, x_n have a one-to-many relationship to the observations o , meaning that the same state can have an infinite number

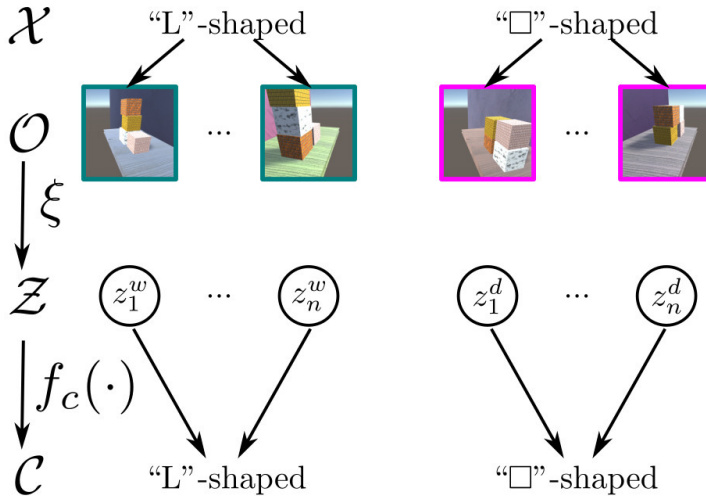


Figure 2.2: An illustrative example of hidden states (“L”-shaped, “□”-shaped) emitting a number of visually different observations. The mapping ξ is used to obtain the representations z , the downstream classifier $f(\cdot)$ is subsequently used to determine the class of the representations (mapping many representations to the same underlying state).

of unique observations. Naturally, we would want to design the representation mapping ξ to have the inverse, a many-to-one relation (many different observations o get mapped to the one true underlying representation z). This relationship is common for state classification tasks. An illustrative example is shown Fig. 2.2, where the arrangement of the boxes constitute the hidden state. Each state emits several visually different observations that then get mapped from observation space \mathcal{O} to the representation space \mathcal{Z} using the mapping ξ . A downstream classifier $f(\cdot)_c$ is then subsequently used to determine the class of a given representation, performing a many-to-one mapping.

As we can see from the example above the performance of a downstream learner $f(\cdot)_c$ depends heavily on the representation space \mathcal{Z} and therefore on the mapping ξ . In principle the more *expressive* the downstream learner $f_c(\cdot)$, the fewer requirements are set on the mapping ξ . As a concrete example if ξ maps the observation to \mathcal{Z} in such a way that the representations are linearly separable, the downstream learner only needs to be a simple linear classifier, while if it maps it to a less-favored *structure* the downstream learner needs to be more expressive. Fig. 2.3 illustrates this example were two different mappings ξ_1 and ξ_2 map the same observations into different representation spaces \mathcal{Z}_1 and \mathcal{Z}_2 respectively. We can easily see that \mathcal{Z}_1 is linearly separable, while \mathcal{Z}_2 is not. An expressive enough downstream learner can solve the task given any of the representations, but a weaker, strictly linear classifier can only succeed in the case of \mathcal{Z}_1 . We can therefore summarise the benefits of

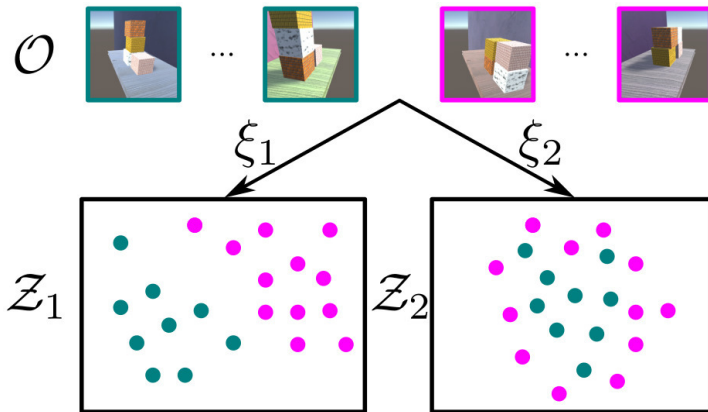


Figure 2.3: Example of two different mappings ξ_1 and ξ_2 map the same observation into different representation spaces. The resulting representation space \mathcal{Z}_1 is linearly separable while \mathcal{Z}_2 is not.

structure in the representations as follows:

The more structured a representation is the less expressive a downstream learner needs to be in order to fulfill the same objective.

Given the benefits of structure in the representation space we add a fifth desired property to the list [36] of desired properties for representations:

- Representations should be well structured.

Given the full list of desired representation properties we can formulate our goal of learning structured representations for object manipulation as follows:

Goal 1. *The goal is to find a mapping $\xi : \mathcal{O} \rightarrow \mathcal{Z}$ that maps $o \in \mathcal{O}$ containing the underlying state x to a structured representation $z \in \mathcal{Z}$ such that it is possible to construct a simple agent $f(\cdot)$ that fulfills a given downstream task by transitioning the system from the current observation to the goal.*

Dividing the Problem

Given the goal formulation, we can divide the overarching problem into three sub-problems that can then be addressed separately and later be combined to present a complete framework.

The three sub-questions addressed are:

- How to build a mapping ξ such that we achieve representations exhibiting the desired properties?
- How to leverage the obtained representations to fulfill the task?

- How to evaluate obtained structured representations and how to compare full frameworks?

The next sections will address these questions and highlight the publications associated with them. In detail, section 2.3 will address the question about how to build the mapping ξ and section 2.4 will discuss the evaluation of representations and full frameworks. Finally we present the Latent Space Roadmap in section 2.5, a combined framework for learning structured representation for rigid and deformable object manipulation.

2.3 Feature Extraction and Structuring Representations

When building a mapping ξ to encode given observations into a suitable representation space \mathcal{Z} , one often employs a *pretext task*. A pretext task is an pre-defined tasks for networks to learn that forces them to produce representation that are more general in nature by optimising for the tasks objective. The intuitive reasoning for employing pretext tasks is that while the task itself is arbitrary the network needs to construct general representation to solve it. These representations can have certain structures or properties that can be exploited by a downstream task, in which we are actually interested.

A representation learning pretext task can be a specific supervised task, such as object classification [29] and simply taking an intermediate layer as a representation, or a generative model relying on reconstruction and KL-Divergence losses as presented in [17]. Other pretext tasks are more arbitrary and are constructed for the sole purpose of later extracting the obtained representation, such as solving a jigsaw puzzle [37], motion segmentation [38], or instance discrimination [39], the latter being the basis for the recent success of the state of art unsupervised visual representation learning methods [40], [41], [42].

In our work in [43] (included as paper A) extending our earlier work [44], we employed a standard VAE. It is based on the idea that similar-looking 2D objects should be encoded close to each other in the model’s representation space. The goal is to obtain representations that preserve their closeness from observation space to representation space. Since computing the euclidean distance between the 128 dimensional representation is much faster than computing the Hamming Distance between the original images and query image, the use of representations makes the cage acquisition procedure viable for real-world settings.

One of the most appealing aspects of many of the mentioned representation learning pretext tasks is that they are completely unsupervised. Furthermore, the current state-of-the-art unsupervised representation learning methods [40], [45], [41], [46], [42], [47], [48], [49], [50] have been closing the gap between the representation obtained from the fully supervised models. It is however important to note that the supervised models are trained not on the instance discrimination task but on a classical multi-class classification task. The way the unsupervised models are able to achieve such useful representation despite not having access to the exact

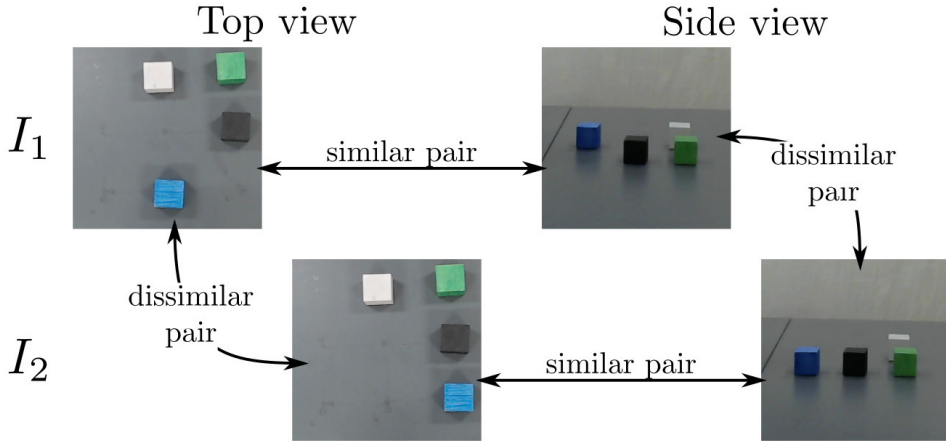


Figure 2.4: Example of *similar* and *dissimilar* pairs. Here, *similar* pairs are obtained by capturing the same scene from two different views, while *dissimilar* pairs are obtained by performing an action (moving the blue box).

class labels is by generating their own labels for the given pretext tasks. For the case of instance discrimination one considers that every image instance is different from any other image in the given dataset, which constitutes *dissimilar* pairs. In order to obtain *similar* pairs, one uses different transformations of the same image.

This is however, not directly applicable in the robotic context. As described in Section 2.2, we can have significantly different observations of the same underlying state and, most importantly, the robotic agent can interact with the environment. We can leverage these task priors in order to obtain supervisory signals in a similar way as in the unsupervised representation learning setting. For dissimilar pairs, we can perform an action that changes the state and, as we know what action we performed, we can record the observation before and after the action as a *dissimilar* pair. For the similar pairs we also have a couple of options, if we are in a setting where task-irrelevant factors change fast, for example, a ceiling fan changes lighting conditions, we can simply wait a certain time between recording observations. Furthermore, if the task has *reversible actions*, meaning applying the inverse of an applied action brings the system to the previous state, we can apply an action and its inverse to obtain different observations from the same underlying state. If multiple viewpoints or an actuated camera is available, we can leverage this to collect different views of the same underlying states in order to generating more *similar* pairs.

In our work in [51] we use given task priors in order to collect *similar* and *dissimilar* pairs in a self-supervised manner for a box manipulation task as shown in Fig. 2.4. The similar pairs were collected by *swapping* boxes with each other, as the underlying state was the arrangement of the boxes. In this way, the robot

could collect the entire dataset in a self-supervised manner. We also established the usefulness of generating additional *dissimilar* pairs by randomly sampling from the full dataset, even if the number of different states is not very high (i.e. ≈ 12). Models that leverage this similar/dissimilar information without relying on any other type of loss exhibited the most structured representations.

In summary, when aiming for structured representations one wants to have *similar* pairs that are as visually different as possible while still being semantically similar, as well as *dissimilar* pairs that are visually similar but semantically dissimilar. When employing task priors and the fact that the agent can actively change its environment and therefore the state of the system, further structuring of representations becomes possible.

2.4 Evaluation of Frameworks and Structured Representations

Comparing methods and/or frameworks with each other in a fair and unbiased way is paramount to the continuous improvement of the robotic research field. When dealing with representation based methods there are two principal options:

1. To compare full resulting framework on standardized task(s).
2. To evaluate the representation directly.

The first option is not only restricted to the case that uses representations in their framework, but also aims to compare a wider range of methods/frameworks on a representative benchmark task establishing clear rules and defined scoring functions. Indeed the prevalence of classification benchmark datasets in the field of computer vision such as the well-known CIFAR-10 [52] and ImageNet [53] datasets for object classification, or the Coco [54] and the PASCAL-VOC [55] dataset for segmentation, shows that the community makes heavy use of standardized benchmark datasets that are accessible to anyone. In the field of robotics, making a dataset or a benchmark has additional challenges, as the goal of most robotic frameworks is to perform a specific task in a limited real-world setting. A method that works for one specific task might be unsuitable in another context. For example, when developing grasping methods the success will heavily depend on the objects one wants to grasp. A big step towards standardizing and making approaches more comparable was the YCB Object and model dataset [56] which is a standardized set of objects (mostly rigid) that is available to order on the author’s website. In this way, different labs can benchmark their methods on the same set of objects making them much more comparable.

In our work in [57] (included as paper C), we followed the YCB benchmark outlaid standards and produced a clear protocol for three different tasks considering typical deformable object manipulation for cloth. We used widely available and standardized objects as well as laid out a precise way to score and compare different approaches.

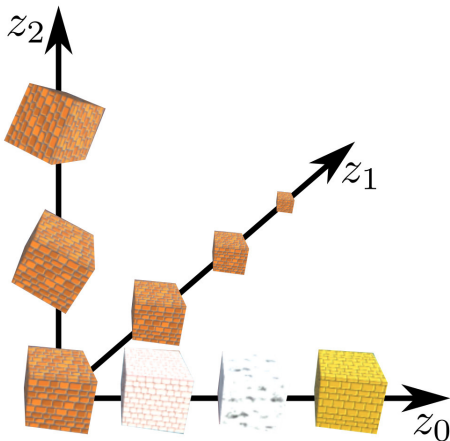


Figure 2.5: Example of disentangling factors of variation, each latent code encodes a specific factor of variation. Here z_0 encodes the color of the box, z_1 encodes the scale, and z_2 encodes the orientation.

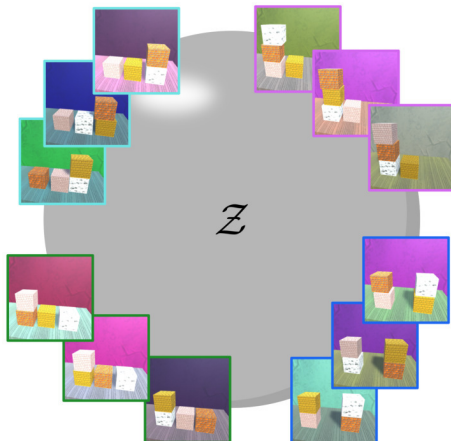


Figure 2.6: Example of a representation space in the shape of a hypersphere, Pushing different classes away from each other on a hypersphere makes classification with a linear classifier possible. Figure inspired by [58].

Evaluating representation directly, on the other hand, has its own challenges. By definition, the representations are supposed to be useful for a certain task or range of tasks, so finding surrogate scores is not a trivial endeavor. One approach that got popularised in [34] is considering the notion of *disentanglement*.

The basic notion of disentanglement is illustrated in Fig. 2.5, where three latent codes (z_1 , z_2 , z_3) are associated with three underlying factors of variation (color, scale, and orientation). A representation is disentangled if changing a single latent code results in a change of a single factor of variation i.e. one independent factor of variation or an underlying generative factor is associated with exactly one latent code, as defined by the authors in [34]. Over the following years a number of disentanglement scores have emerged [59], [60], [61], [62], [63], [64], [65]. However, there is currently no agreement in the community about what score serves best as a surrogate score to evaluate representations. In [66] the authors challenge a number of common assumptions regarding unsupervised learning of disentangled representations and demonstrate that the random initialization and model parameters can have a much larger impact than the different models used for learning disentangled representation. While there has been some works leveraging disentangled representation in a reinforcement learning context [67], on the topic of Sim2real transfer [68], and for predicting out-of-distribution (OOD) performance [69], the benefits in a real-world robotic manipulation context are currently not established.

A more agreed-upon evaluation protocol in visual unsupervised representation

learning is to evaluate the representations by the accuracy of a simple linear classifier [40], [45], [41], [46], [42]. The idea is that a representation that facilitates good classification results for a simple linear classifier is more useful than one that needs a more complex or expressive downstream learner. For example, [58] provides an in-depth analysis on normalizing the representation space onto a hypersphere, where a linear classifier is able to perform well if classes are correctly clustered. This concept is sketched in Fig. 2.6, where different observation instances are clustered together, and the different classes are uniformly distributed throughout the hypersphere. The caveat of this method, however, is that it requires ground truth state labels, while they are readily available in computer classification tasks, or in simulated settings, they are seldom present in more complex robotic manipulation scenarios especially when dealing with deformable objects.

In our work in [51] (included as paper E), we were specifically interested in evaluating the structure of the representations, as well as their utility in a robotics visual planning task. Therefore, we propose a two-pronged approach:

i) As the structure we are interested in is very closely related to *clusterability*, we perform a number of cluster related scores such as recording the number of clusters, the homogeneity and completeness [70], as well as the mean silhouette coefficient [71], whereof the latter does not rely on ground truth labels.

ii) The planning performance is evaluated on our Latent Space Roadmap (LSR) framework [72] (paper X-3, not included) and the assumption that all potential actions are always correct. With this setup, we compared the effect of different kinds of loss functions. Namely we showed that a contrastive loss is more beneficial in the context of visual task planning in robotics than a reconstruction based one.

2.5 Latent Space Roadmap - An Example of a Combined Framework

Our work presented in [73] (included as paper B, currently under review) which is an extended version of our work presented in [72] (not included paper X-3) gives an example of a combined framework that learns representations and then employs them in order to perform rigid and deformable object manipulation. An overview of the framework is shown in Fig. 2.7, which consists of three components:

i) A Mapping Module ξ , which takes a start and goal observation and maps them into the latent space \mathcal{Z} . We realized this map ξ with a VAE where we added an additional contrastive loss term (called action loss in [73]) that makes use of the manually obtained similar/dissimilar pairs information in order to produce a more structured representation. We established that adding a contrastive term improves the structure of the latent space in our work in [51] (included as paper E (under review)), however an important point in the provided framework is that we produce visual action plans, and we therefore decode any given latent representation with a decoder ω .

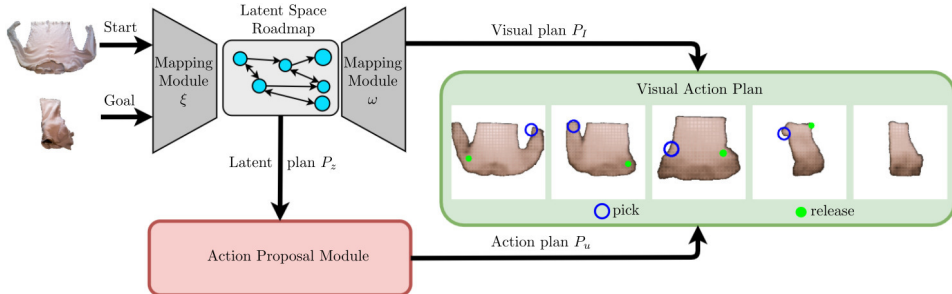


Figure 2.7: Overview of our LSR framework, consisting of the Mapping Module ξ , the Latent Space Roadmap (LSR), and the Action Proposal Module. The framework takes in a start and goal observation and produces a visual action plan that contains the intermediate observations and actions needed to reach the desired goal configuration (Figure is excerpt from our work in [73]).

ii) The Latent Space Roadmap (LSR) which is a graph built directly inside the latent space where the nodes ideally represent the underlying states and the edges represent possible transitions. It is constructed in three phases: *a)* in the first phase a reference graph is constructed using the information if a pair in the dataset is a similar pair (no edge is built between them) or a dissimilar pair (an edge is built). *b)* For the second phase, the latent space is clustered using agglomerative clustering, we measure the distance of inter-cluster dissimilarity using the unweighted average distance between points in the considered clusters. The final clustering is then obtained by applying the clustering threshold τ to the obtained stepwise dendrogram, as shown in Fig. 2.8 top right. *c)* In the last phase, the LSR is built where each cluster obtained in the previous phase corresponds to a node in the LSR, and they are connected if there is an edge between the individual pairs (as obtained in phase a) located inside different clusters. We can have an optional additional pruning step that removes nodes that only have a certain number of data points as members or edges that have less than a specified number of edges connecting the same nodes. This step can be helpful if the collected training data is noisy and contains outliers.

The attentive reader might have noticed that the performance of the clustering heavily depends on the clustering threshold τ , which is not trivial to tune and subject to ongoing research [74], [75]. For this reason, we propose an outer optimization loop that substitutes the hard-to-tune τ hyperparameter with a much more robust and easier to tune parameter c_{max} . In our outer optimization, we analyze the number of edges as a measurement of LSR quality and optimize for a τ that gives the most edges in the LSR while having fewer than c_{max} separate graph connected components. The intuitive motivation is that we want to have an LSR that is well connected, and therefore should have many edges. But we also want to have only a limited number of connected components as this lowers the connectivity significantly without affecting the total number of edges. When optimizing

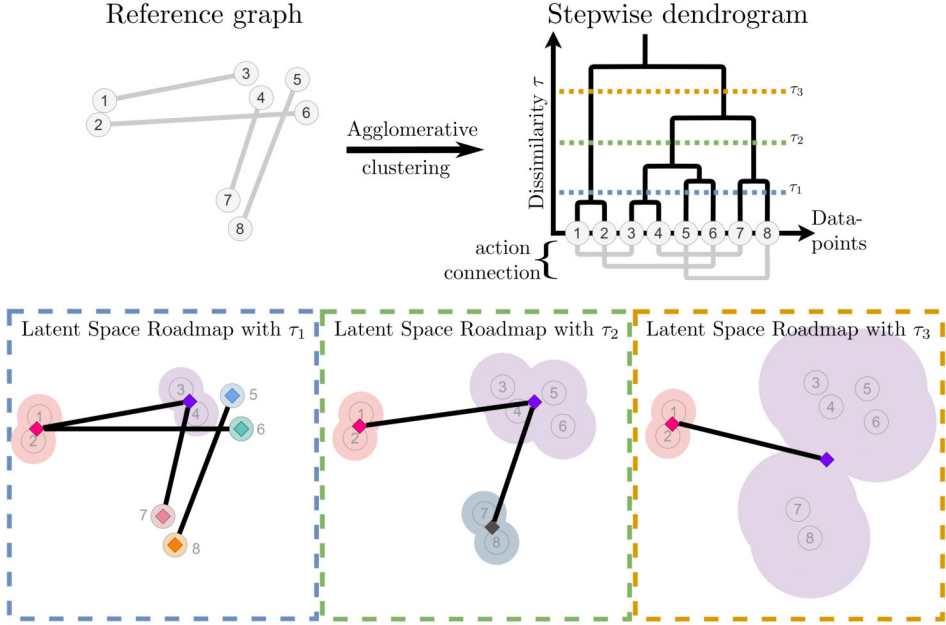


Figure 2.8: Illustration of the outer optimisation for the clustering threshold τ . A reference graph (top left) results in a stepwise dendrogram after applying agglomerative clustering. The different thresholds τ_1 , τ_2 , and τ_3 results in different LSRs. (Figure is excerpt from our work in [73])

with a set hyperparameter we can obtain an LSR that is very well connected and consists of at least one large connected component. When employing a very low τ the LSR receives more edges but a large number of disconnected components, while if τ is large it exhibits few disconnected components but also few edges. Fig. 2.8 shows this principle. On the top row we see the reference graph and the resulting stepwise dendrogram. This dendrogram can then be cut by the clustering threshold τ resulting in different LSRs (shown on the bottom row). The diamond markers indicate separate nodes while the edges are visualized with connecting black lines. We can see that the left most LSR (τ_1) has the most edges but also two connected components, while the τ_2 and τ_3 result in a single connected component where τ_2 obtains more edges.

iii) The final part of the framework is the action proposal model, which takes as input a latent plan, produced by the LSR, and proposes actions to translate from one state to the next. In our case, we employed a Multilayer Perceptron network trained on the representations obtained with the mapping ξ . Note that it is also possible to integrate the action proposal directly into the LSR, for example, a simple baseline action can be obtained by averaging the actions associated with each edge

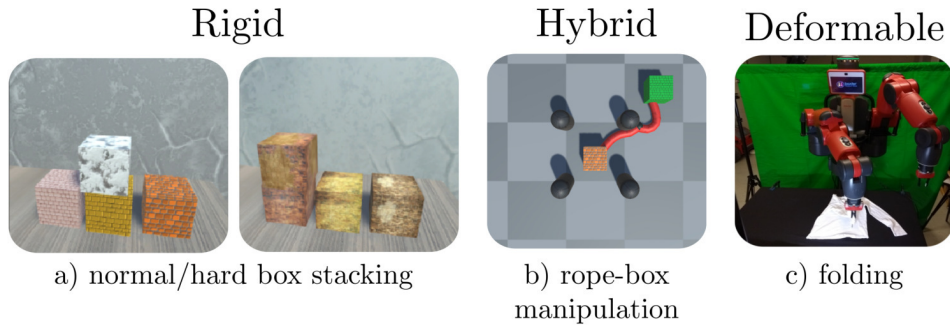


Figure 2.9: Different tasks addressed with the LSR framework. A rigid box stacking task in two variants shown in a), a hybrid rope-box manipulation task shown in b), and a folding task on a highly deformable object in c).

in the reference graph into the corresponding edges in the LSR.

Overall the framework uses these three components to obtain visual action plans that can be executed on a real robotic system. This framework works best in the setting where the feasible states of the system are finite and can be distinguished from each other in such a way that unambiguous actions to transition between them can be defined. We extensively evaluated and showcased the performance of this framework on three different tasks (shown in Fig. 2.9), namely a rigid box stacking task, a hybrid rope-box task where two boxes are connected with a rope, and a real-world T-shirt folding task.

Chapter 3

An Overview of Publications

A general overview of the published papers (included and not-included in this thesis) is shown in Fig. 3.1. This will serve as an overall orientation, highlighting the links between different publications and how they contributed by addressing the specific questions asked in this thesis. Papers that are not included in this thesis are shown with a purple background and marked with X-1 to X-5, while works included are appearing with a green background and labeled with the letters A-E.

Paper X-1 “From visual understanding to complex object manipulation” [76] gave an overview of the whole manipulation pipeline that is required to successfully manipulate objects. The process starts with a visual step where relevant objects and obstacles need to be identified, it continues with a planning step taking into account the environment as well as the constraints imposed by the capabilities of the manipulator, the follow-up step is sensor feedback where several potential sources of information like tactile readings or visual feedback can be considered. The final step is then the manipulation itself, where the grasped object is put into the desired configuration or employed as a tool. The project gave a good overview of the challenges to overcome when realizing such a system in the real world as well as insights into each step required to successfully perform object manipulation.

One part that was of tangential interest was how a real-world vision system specialized for cloth manipulation could be realized and how to address the specific scene encountered in robotic cloth manipulation. The resulting publication X-4 “Fashion Landmark Detection and Category Classification for Robotics” [77] presents an elastic warping method to augment training data found in large fashion datasets to make them more relevant for a robotic context. It furthermore investigated how robust landmark detection is under occlusion from a robotic arm.

Paper A “Partial caging: a clearance-based definition, datasets and deep learning” [43] is the extended version of paper X-2. This paper is directly relevant for the thesis as we are using a VAE trained on known objects to then encode novel objects, select the closest known objects in the latent space and propose high-quality partial cages inspired by the known object for the novel object. Here the latent space generated by the VAE was used in order to facilitate a different downstream

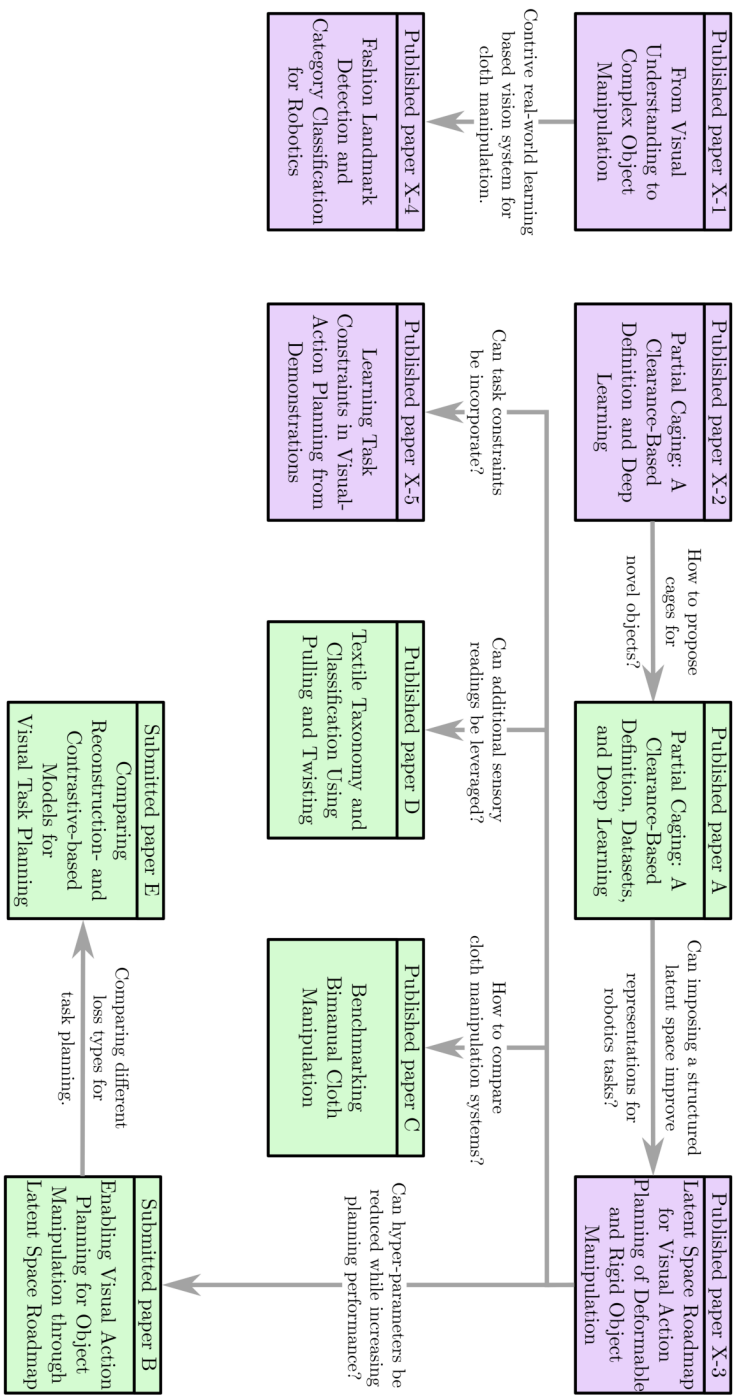


Figure 3.1: Overview of publications. PhD started with the two projects in the top left corner. The arrows symbolise high level questions that were explored and led to the subsequent publications. Items with green background are included in this thesis while items with purple are not.

task (partial cage acquisition). Our findings however showed that while the use of representation was significantly faster compared to the Hamming distance from the novel object to any other known object, the quality of the proposed partial cages was higher using the Hamming distance compared to using the representation of the VAE. This showed that while representations are a promising direction one needs to introduce more structure that shapes the latent representation to a more favorable state given the downstream tasks.

The lessons learned from paper X-1 regarding the real-world challenges and the usefulness of representations for downstream tasks from paper A laid the groundwork for paper X-3 “Latent Space Roadmap for visual action planning of deformable and rigid object manipulation” [72]. In this work, the core question was how to learn structured representation as well as how to leverage it into a real-world system that is able to complete challenging robotic tasks like folding. We tackled the problem of visual action planning where given a start and goal observation a visual action plan is produced that shows the planned steps as well as the actions required to traverse from the start state to the desired goal state. We leveraged the task priors that, when successfully performing an action, the subsequent state is different from the initial one but the same if only minor changes are performed that do not change the underlying state. This knowledge is leveraged when generating the representation space using a VAE. We augmented the loss function of the VAE to include a contrastive term that pushes pairs of states that are dissimilar apart and contracts pairs of states that are similar. Using this representation the LSR was built in the latent space by first performing clustering and then connecting the cluster with edges representing the possible actions observed in the training dataset. We validated our LSR method on a simulated box stacking and real-world shirt folding task. While this project addresses the core of the thesis, there were many open questions: *i)* how to best achieve the mapping from observation into representation space? Is a VAE the only option? *ii)* Which clustering method for building the LSR is most suited? Can the clustering hyperparameter be automatically tuned? *iii)* How does the method compare to other methods? What are the limitations of the framework and can it perform on tasks where rigid and deformable objects are combined? This questions directly relates to the LSR and are subsequently addressed in paper B. However, there were also more fundamental questions arising that are also relevant for the overall topic of this thesis. Namely, *iv)* how can one compare different cloth manipulation systems? *v)* Can additional sensory readings be leveraged to gain more insights about the clothing items? And *vi)* can specific task constraints be incorporated into the framework?

Question *i-iii)* were answered in paper B “Enabling Visual Action Planning for Object Manipulation through Latent Space Roadmap” [73], where the framework is split into three distinct parts, a Mapping Module that can be realized with any encoder like model (we compared AE with VAE), the LSR built with a new core clustering algorithm (unweighted average linkage) as well as the inclusion of an outer optimization loop with easier-to-tune hyperparameters, and finally the Action Proposal Model realizable with either simple average action aggregations

or any neural network. Furthermore, an additional hard box stacking task was introduced to complement the normal box stacking task and make the ablation study more relevant and wider in scope. A new hybrid task, where two boxes are connected with a deformable rope is also added. In this work, we showed extensively how a system that learns structured representation is used for simulated rigid and real-world deformable object manipulation.

Our work presented in paper C “Benchmarking Bimanual Cloth Manipulation” [57] takes a step towards resolving question *iv*), of how to compare different methods/systems. We proposed a benchmark featuring three standardized tasks, spreading a tablecloth, folding a towel, and a dressing task. The tasks were designed in such a way that they are easily reproducible and come with a consistent scoring protocol that allows to benchmark and compare different approaches against each other. Defining standardized tasks in a reproducible and comparable way is an important step to make it easier to evaluate and compare different methods. It also highlights the strengths and weaknesses in a standardized manner such that future research can be easily extended and improve existing approaches.

When dealing with deformable objects any information the system can obtain thanks to additional sensory readings might be of great benefit. In the specific case of cloth manipulation, inferring the dynamic behavior by knowing something about the material or construction techniques could be of significant benefit. In our work presented in paper D “Textile Taxonomy and Classification Using Pulling and Twisting” [78] we take a step towards this direction. First we introduced a taxonomy that not only considers the material of garments but also the constructing technique such as if it is woven or knitted. We postulated that the construction technique plays a significant factor in the dynamics of garments that have not been considered in prior work. In a small-scale experiment, we demonstrated that one can infer the construction technique with higher accuracy than the material using force/torque readings and pulling actions.

The final question, whether specific task constraints can be incorporated, was considered in our work published in paper X-5 “Learning Task Constraints in Visual Action Planning from Demonstration” [79], where we used Linear Temporal Logic (LTL) formulation to label demonstrated sequences that either fulfill the given constraint or not. An Long short-term memory (LSTM)-model [80] was then used to distinguish between the binary case. Different LTL-formulated constraints can be acquired this way and then combined with simple logical operators such as AND or OR to combine simple constraints into more complex ones.

While we performed an extensive ablation study in paper B, the nature of developing a full system with many components is that one does not have the space to extensively analyse every aspect of such a framework. While we showed that a VAE is a more suitable mapping module than an AE for the tasks considered, we did not perform a fundamental analysis that compared models based on the *type* of loss function they employ and their usefulness for higher-level planning tasks. This is addressed in paper E “Comparing Reconstruction-and Contrastive-based Models for Visual Task Planning” [51] (currently under review) where we first analyzed

an extensive body of related work and discovered that despite the introduction of contrasting losses in 2006, many current state-of-the-art methods and proposed frameworks still rely on the reconstruction loss. We performed a systematic study comparing seven different models that either employed pure reconstruction loss, reconstruction- and KL-loss, pure contrastive loss, or a combination of the three different loss types. We evaluated the models on three different visual planning tasks where irrelevant factors of variations are present in the observations given to the system. We showed that purely contrastive-based losses are best suited to address such tasks especially if observations of the same underlying state can significantly differ from each other because of task-irrelevant factors such as distractor objects, or different view angles.

Chapter 4

Conclusion and Future Work

In this thesis, we looked at how to learn representation for rigid and deformable object manipulation. We presented our work and the contribution on learning structured representations for rigid and deformable object manipulation, concluding with a framework incorporating the lessons learned that was successfully applied to tasks from the rigid, hybrid, and deformable object manipulation domain. To conclude this thesis we will shortly highlight potential future work that expands the lessons learned and try to uncover more relevant insights.

4.1 Cage-Flow - A Flow-based Cage Proposal Model

The mapping $\xi : \mathcal{O} \rightarrow \mathcal{Z}$ considered so far was predominantly modeled using a VAE which optimizes the evidence lower bound (ELBO). There exists however a different group of likelihood-based generative models - namely normalizing flow models [81], [82], [83]. In normalizing flow models, the goal is to map a simple distribution (one that can easily be sampled from) to a more complex one (inferred from the data). A core difference to VAEs is that the mapping from observation space \mathcal{O} to the representation space \mathcal{Z} is deterministic and invertible. A flow-based model therefore directly approximates the distribution of the given training data. Flow-based models have been successfully applied in many domains like video generation [84], graph generation [85], and reinforcement learning [86], [87].

In our future work, we want to employ normalizing flow model to predict promising cages for 2D and 3D objects, as well as investigate the estimated distributions. Another interesting aspect we want to consider is that cages can be represented in different ways. Therefore, an additional goal is to examine what kind of cage representations is most suitable for flow-based models. In order to achieve this goal, we produce a number of *promising* cages on randomly generated objects (so far only 2D), by employing a number of handcrafted heuristics such as placing a caging tool inside the symmetric difference of the object and its convex-hull, perpendicular to the longest line, on opposite sides of the predominate object axis, and more. We represent these cages as geometric graphs with a central *anchor* node and define

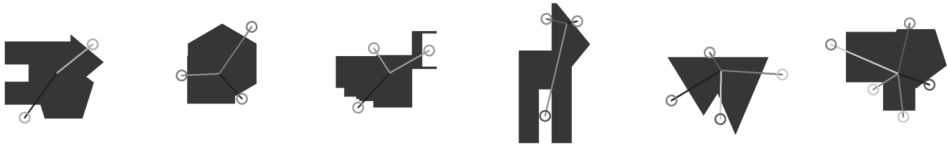


Figure 4.1: Example of 2D cages, represented as a graph having a center *anchor*-node. The cages were obtained employing a number of handcrafted heuristics.

the location of the caging points using polar coordinates. Another potential representation is a simple set of points in planar coordinates. Fig. 4.1 shows some examples of cages obtained using the mentioned heuristics with varying amounts of caging points.

Our next step is to train a conditional flow-based model on either the graph structure or the set of points representations while conditioning it on the object to be caged. In detail, we plan to investigate the following questions:

1. Can promising caging configuration for novel objects be obtained for 2D and 3D objects using Cage-Flow?
2. How does the estimated distribution change when adding more objects or caging tools to the training data?

4.2 Hierarchical-LSR - A LSR framework for Continuous Problems

The current LSR framework has been successfully applied to different domains and a number of diverse tasks. It works best in the setting of higher-level planning employing action primitives, and considering systems that have a *relatively* low number of clearly separated underlying states. We are planning to expand this framework also to tasks of a more *continuous* nature. The problem of separating states can be seen as a discretization problem with multiple layers, where the higher level corresponds to the current application of the LSR. A potential approach would be to develop a hierarchical approach where states are further discretized, for example, the highest level describes the arrangement of boxes while at a lower level the state descriptor identifies the rotation or exact position. Such a framework could be employed to solve a wider range of tasks than the current one presented in this thesis.

Furthermore, we are planning to investigate how to integrate the LSR in a reinforcement learning setting. We plan to build an initial LSR and based on this decide what kind of states transitions are needed, actively explore those, and extend the LSR step by step. Such an approach could also be leveraged to find *shortcuts* between states, currently, we can only transition between states with actions that have been observed in the training data, however, there are some problem settings

where the action observed between two different states could also be generalized onto a different pair of states. An LSR explorer could attempt to traverse from one state to another with the most promising action and extend the connectivity of the LSR in this way.

In conclusion, this thesis provided a step towards learning structured representations for rigid and deformable object manipulation, by answering relevant research questions as well as advancing the state of the art with the new Latent Space Roadmap method.

Bibliography

- [1] Y. Huang, M. Bianchi, M. Liarokapis, and Y. Sun, “Recent data sets on object manipulation: A survey,” *Big data*, vol. 4, no. 4, pp. 197–216, 2016.
- [2] L. Jing and Y. Tian, “Self-supervised visual feature learning with deep neural networks: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [3] H. Yin, A. Varava, and D. Kragic, “Modeling, learning, perception, and control methods for deformable object manipulation,” *Science Robotics*, vol. 6, no. 54, 2021.
- [4] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik *et al.*, “Sofa: A multi-model framework for interactive physical simulation,” in *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012, pp. 283–321.
- [5] X. Lin, Y. Wang, J. Olkin, and D. Held, “Softgym: Benchmarking deep reinforcement learning for deformable object manipulation,” 2020.
- [6] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
- [7] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2021.
- [8] “Nvidia, physx sdk,” <https://developer.nvidia.com/physx-sdk>, accessed: 2021-09-29.
- [9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013, cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. [Online]. Available: <http://arxiv.org/abs/1312.5602>
- [10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

- [11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2016.
- [12] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, “Hindsight experience replay,” 2017.
- [13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms.” *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: <http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17>
- [14] S. Levine, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search,” *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 156–163, 2015.
- [15] J. Matas, S. James, and A. J. Davison, “Sim-to-real reinforcement learning for deformable object manipulation,” in *Conference on Robot Learning*. PMLR, 2018, pp. 734–743.
- [16] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller, “Embed to control: A locally linear latent dynamics model for control from raw images,” 2015.
- [17] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *CoRR*, vol. abs/1312.6114, 2014.
- [18] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [19] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, “Learning latent dynamics for planning from pixels,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 2555–2565.
- [20] R. Y. Rubinstein, “Optimization of computer simulation models with rare events,” *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.
- [21] K. Chua, R. Calandra, R. McAllister, and S. Levine, “Deep reinforcement learning in a handful of trials using probabilistic dynamics models,” 2018.
- [22] B. Ichter and M. Pavone, “Robot motion planning in learned latent spaces,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.
- [23] D. H. Ballard, “Modular learning in neural networks.” in *AAAI*, vol. 647, 1987, pp. 279–284.

- [24] N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-parametric topological memory for navigation,” 2018.
- [25] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.
- [26] S. Nair and C. Finn, “Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation,” *International Conference on Learning Representations (ICLR)*, 2020.
- [27] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, “Learning robotic manipulation through visual planning and acting,” *Robotics: Science and Systems (RSS)*, 2019.
- [28] M. Yan, Y. Zhu, N. Jin, and J. Bohg, “Self-supervised learning of state estimation for manipulating deformable linear objects,” *IEEE robotics and automation letters*, vol. 5, no. 2, pp. 2372–2379, 2020.
- [29] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2015.
- [30] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, “Spatial transformer networks,” *Advances in neural information processing systems*, vol. 28, pp. 2017–2025, 2015.
- [31] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, “Learning predictive representations for deformable objects using contrastive estimation,” *CoRL*, 2020.
- [32] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *CoRR*, vol. abs/1807.03748, 2018. [Online]. Available: <http://arxiv.org/abs/1807.03748>
- [33] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, “VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation,” 2020.
- [34] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [35] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, “State representation learning for control: An overview,” *Neural Networks*, vol. 108, pp. 379–392, 2018.
- [36] W. Böhmer, J. T. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer, “Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement

- learning agents from their real-world sensor observations,” *KI-Künstliche Intelligenz*, vol. 29, no. 4, pp. 353–362, 2015.
- [37] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *European conference on computer vision*. Springer, 2016, pp. 69–84.
- [38] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, “Learning features by watching objects move,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2701–2710.
- [39] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.
- [40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [41] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.
- [42] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” 2020.
- [43] M. C. Welle, A. Varava, J. Mahler, K. Goldberg, D. Kragic, and F. T. Pokorny, “Partial caging: a clearance-based definition, datasets, and deep learning,” *Autonomous Robots*, pp. 1–18, 2021.
- [44] A. Varava, M. C. Welle, J. Mahler, K. Goldberg, D. Kragic, and F. T. Pokorny, “Partial caging: A clearance-based definition and deep learning,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1533–1540.
- [45] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, “Big self-supervised models are strong semi-supervised learners,” 2020. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/fcbc95ccdd551da181207c0c1400c655-Abstract.html>
- [46] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, “Bootstrap your own latent - a new approach to self-supervised learning,” vol. 33, pp. 21 271–21 284, 2020. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf>

- [47] M. Kang and J. Park, “ContraGAN: Contrastive Learning for Conditional Image Generation,” *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [48] T. Xiao, X. Wang, A. A. Efros, and T. Darrell, “What should not be contrastive in contrastive learning,” 2021. [Online]. Available: <https://openreview.net/forum?id=CZ8Y3NzuVzO>
- [49] X. Chen and K. He, “Exploring simple siamese representation learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 750–15 758.
- [50] M. Laskin, A. Srinivas, and P. Abbeel, “Curl: Contrastive unsupervised representations for reinforcement learning,” *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020, arXiv:2004.04136.
- [51] C. Chamzas, M. Lippi, M. C. Welle, A. Varava, L. E. Kavraki, and D. Kragic, “Comparing reconstruction-and contrastive-based models for visual task planning,” *arXiv preprint arXiv:2109.06737*, 2021.
- [52] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” no. 0, 2009.
- [53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [55] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results,” <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [56] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The ycb object and model set: Towards common benchmarks for manipulation research,” in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.
- [57] I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Alenya *et al.*, “Benchmarking bimanual cloth manipulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1111–1118, 2020.

- [58] T. Wang and P. Isola, “Understanding contrastive representation learning through alignment and uniformity on the hypersphere,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 9929–9939.
- [59] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework,” 2017.
- [60] H. Kim and A. Mnih, “Disentangling by factorising,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 2649–2658.
- [61] A. Kumar, P. Sattigeri, and A. Balakrishnan, “Variational inference of disentangled latent concepts from unlabeled observations,” *CoRR*, vol. abs/1711.00848, 2017. [Online]. Available: <http://arxiv.org/abs/1711.00848>
- [62] T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, “Isolating sources of disentanglement in variational autoencoders,” 2018. [Online]. Available: <https://openreview.net/forum?id=BJdMRoCif>
- [63] C. Eastwood and C. K. Williams, “A framework for the quantitative evaluation of disentangled representations,” in *International Conference on Learning Representations*, 2018.
- [64] K. Ridgeway and M. C. Mozer, “Learning deep disentangled embeddings with the f-statistic loss,” 2018.
- [65] R. Suter, D. Miladinovic, B. Schölkopf, and S. Bauer, “Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 6056–6065.
- [66] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, “Challenging common assumptions in the unsupervised learning of disentangled representations,” in *international conference on machine learning*. PMLR, 2019, pp. 4114–4124.
- [67] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, “Darla: Improving zero-shot transfer in reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1480–1490.
- [68] M. W. Gondal, M. Wuthrich, D. Miladinovic, F. Locatello, M. Breidt, V. Volchkov, J. Akpo, O. Bachem, B. Schölkopf, and S. Bauer, “On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset,” in *NeurIPS*, 2019, pp. 15 714–15 725.

- [69] A. Dittadi, F. Träuble, F. Locatello, M. Wuthrich, V. Agrawal, O. Winther, S. Bauer, and B. Schölkopf, “On the transfer of disentangled representations in realistic settings,” in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=8VXvj1QNRl1>
- [70] A. Rosenberg and J. Hirschberg, “V-measure: A conditional entropy-based external cluster evaluation measure,” in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410–420.
- [71] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.
- [72] 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,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5619–5626.
- [73] —, “Enabling visual action planning for object manipulation through latent space roadmap,” *arXiv preprint arXiv:2103.02554*, 2021.
- [74] D. Bruzzese and D. Vistocco, “Despota: Dendrogram slicing through a permutation test approach,” *Journal of classification*, vol. 32, no. 2, pp. 285–304, 2015.
- [75] A. Pasini, E. Baralis, P. Garza, D. Floriello, M. Idiomi, A. Ortenzi, and S. Ricci, “Adaptive hierarchical clustering for petrographic image analysis.” in *EDBT/ICDT Workshops*, 2019.
- [76] J. Bütepage, S. Cruciani, M. Kokic, M. Welle, and D. Kragic, “From visual understanding to complex object manipulation,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 161–179, 2019.
- [77] T. Ziegler, J. Bütepage, M. C. Welle, A. Varava, T. Novkovic, and D. Kragic, “Fashion landmark detection and category classification for robotics,” in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2020, pp. 81–88.
- [78] A. Longhini, M. C. Welle, I. Mitsioni, and D. Kragic, “Textile taxonomy and classification using pulling and twisting,” *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.
- [79] F. Esposito, C. Pek, M. C. Welle, and D. Kragic, “Learning task constraints in visual-action planning from demonstrations,” in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 131–138.

- [80] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [81] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.
- [82] I. Kobyzev, S. Prince, and M. Brubaker, “Normalizing flows: An introduction and review of current methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [83] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing flows for probabilistic modeling and inference,” *Journal of Machine Learning Research* 22 (2021) 1-64, 2021.
- [84] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma, “Videoflow: A conditional flow-based model for stochastic video generation,” 2020. [Online]. Available: <https://openreview.net/forum?id=rJgUfTEYvH>
- [85] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe, “Graph{nvp}: an invertible flow-based model for generating molecular graphs,” 2020. [Online]. Available: <https://openreview.net/forum?id=ryxQ6T4YwB>
- [86] B. Mazouze, T. Doan, A. Durand, J. Pineau, and R. D. Hjelm, “Leveraging exploration in off-policy algorithms via normalizing flows,” in *Conference on Robot Learning*. PMLR, 2020, pp. 430–444.
- [87] P. N. Ward, A. Smofsky, and A. J. Bose, “Improving exploration in soft-actor-critic with normalizing flows policies,” *INNF workshop, International Conference on Machine Learning 2019*, 2019.

Part II

Included Publications

Paper A

Paper A

Partial Caging: A Clearance-Based Definition, Datasets, and Deep Learning

Michael C. Welle, Anastasiia Varava, Jeffrey Mahler, Ken Goldberg,
Danica Kragic and Florian T. Pokorny

Abstract

Caging grasps limit the mobility of an object to a bounded component of configuration space. We introduce a notion of partial cage quality based on maximal clearance of an escaping path. As computing this is a computationally demanding task even in a two-dimensional scenario, we propose a deep learning approach. We design two convolutional neural networks and construct a pipeline for real-time planar partial cage quality estimation directly from 2D images of object models and planar caging tools. One neural network, CageMaskNN, is used to identify caging tool locations that can support partial cages, while a second network that we call CageClearanceNN is trained to predict the quality of those configurations. A partial caging dataset of 3811 images of objects and more than 19 million caging tool configurations is used to train and evaluate these networks on previously unseen objects and caging tool configurations. Experiments show that evaluation of a given configuration on a GeForce GTX 1080 GPU takes less than 6 ms. Furthermore, an additional dataset focused on grasp-relevant configurations is curated and consists of 772 objects with 3.7 million configurations. We also use this dataset for 2D Cage acquisition on novel objects. We study how network performance depends on the datasets, as well as how to efficiently deal with unevenly distributed training data. In further analysis, we show that the evaluation pipeline can approximately identify connected regions of successful caging tool placements and we evaluate the continuity of the cage quality score evaluation along caging tool trajectories. Influence of disturbances is investigated and quantitative results are provided.

1 Introduction

A rigid object is *caged* if it cannot escape arbitrarily far from its initial position. From the topological point of view, this can be reformulated as follows: an object is caged if it is located in a bounded connected component of its free space. This notion provides one of the rigorous paradigms for reasoning about robotic grasping besides form and force closure grasps [1], [2]. While form and force-closure are concepts that can be analyzed in terms of local geometry and forces, the analysis of caging configurations requires knowledge about a whole connected component of the free configuration space and is hence a challenging problem that has been extensively studied analytically. However, since global properties of configuration space may also be estimated more robustly than subtle local geometric features used in classical force closure analysis, caging may hold promise particularly as a noise-tolerant approach to grasping and manipulation.

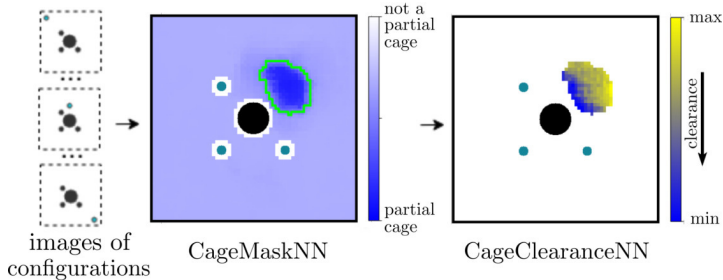


Figure A.1: Given an image of an object (depicted in black) and 3 or 4 caging tools (depicted in green), *CageMaskNN* determines whether a configuration belongs to the “partial cage” subset. If it does, *CageClearanceNN*, evaluates its quality according to the clearance measure learned by the network. On the figure, the blue region corresponds to successful placements of the fourth finger according to *CageMaskNN*, and their quality predicted by *CageClearanceNN*.

In its topological formulation, caging is closely related to another global characteristic of configuration spaces – path-connectedness, and, in particular, is a special case of the path non-existence problem [3, 4]. This is a challenging problem, as it requires reasoning about the entire configuration space, which is currently not possible to reconstruct or approximate [3, 4].

Another interesting global characteristic of a configuration space is the maximum clearance of a path connecting two points. In path planning, paths with higher clearance are usually preferred for safety reasons. In contrast, in manipulation, if an object can escape from the manipulator only through a narrow passage, escaping is often less likely. In practical applications, it might be enough to partially restrict the mobility of the object such that it can only escape through narrow passages instead of completely caging it. Such configurations are furthermore less restrictive than full cages, thus allowing more freedom in placing caging tools.

This reasoning leads to the notion of *partial caging*. This generalization of classical caging was first introduced by Makapunyo et al. [5], where the authors define a partial caging configuration as a non-caging formation of fingers that only allows rare escape motions. While [6] and [7] define a similar notion as energy-bounded caging, we propose a partial caging quality measure based on the maximum clearance along any possible escaping path. This value is directly related to the maximum width of narrow passages separating the object from the rest of the free space. Assuming motion is random, the quality of a partial cage depends on the width of a “gate” through which the object can escape.

Our quality measure is different from the one proposed in [5], where the authors introduced a measure based on the complexity and length of paths constructed by a sampling-based motion planner, thus generalizing the binary notion of caging to a property parameterized by cage quality.

One challenge with using sampling-based path planners for partial caging evaluation is that a single configuration requires multiple runs of a motion planner and – in the case of rapidly exploring random tree (RRT) – potentially millions of tree expansion steps each, due to the non-deterministic nature of these algorithms. This increases the computation time of the evaluation process which can be critical for real-time applications, such as scenarios where cage quality needs to be estimated and optimized iteratively to guide a caging tool from a partial towards a final cage. We significantly speed up the evaluation procedure for partial caging configurations by designing a deep learning-based pipeline that identifies partial caging configurations and approximates the partial caging evaluation function (we measured an evaluation time of less than 6 ms for a single given configuration on a GeForce GTX 1080 GPU). For this purpose, we create a dataset of 3811 two-dimensional object shapes and 19055000 caging tool configurations and use it to train and evaluate our pipeline.

Apart from evaluating given partial caging configurations, we also use the proposed quality measure to choose potentially successful placements of 1 out of 3 or 4 caging tools, assuming the positions of the remaining tools are fixed. In Fig. A.1, we represent the output as a heat map, where for every possible translational placement of a caging tool along a grid the resulting partial caging quality value is computed. Another application of the pipeline is the evaluation and scoring of caging configurations along a given reference trajectory.

Furthermore, we explore different shape similarity measures for objects and evaluate them from the partial caging perspective. We propose a way to generate partial caging configurations for previously unseen objects by finding similar objects from the training dataset and applying partial caging configurations that have good quality score for these objects. We compare three different definitions of distance in the space of shapes: Hausdorff, Hamming, and the distance in the latent space of a variational autoencoder (VAE) trained on a set of known objects. Our experiments show that Hamming distance is the best at capturing geometric features of objects that are relevant for partial caging, while the VAE-induced distance has the advantage of being computationally efficient.

This paper is a revised and extended version of our previously published conference submission [8]. The contribution of the extension with respect to the conference paper can be summarized as follows:

1. we define a *grasping band* for planar objects – the area around the object that is suitable for placing caging tools, created a new dataset¹ consisting of partial caging configurations located in the grasping band;
2. we approximate our partial caging quality measure with a deep neural network trained on this new dataset;
3. we perform ablation studies to evaluate our deep network architecture;
4. we evaluate the adequacy of our partial caging quality measure by modeling the escaping process as a random walk, and measuring the escape time;
5. we propose a cage acquisition method for novel objects based on known partial caging configurations for similar objects; for this, we explore several different distance metrics;
6. we further evaluate the robustness of the cage acquisition with respect to noise.

2 Related Work

One direction of caging research is devoted to point-wise caging, where a set of points (typically two or three) represents fingertips, and an object is usually represented as a polygon or a polyhedron, an example of a 2D cage can be seen in Fig. A.2 on the left-hand side. Rimon and Blake in their early work [9] proposed an algorithm to compute a set of configurations for a two-fingered hand to cage planar non-convex objects. Later, Pipattanasomporn and Sudsang [10] proposed an algorithm reporting all two-finger caging sets for a given concave polygon. Vahedi and van der Stappen in [11] described an algorithm that returns all caging placements of a third finger when a polygonal object and a placement of two other fingers are provided. Later, Rodriguez et al. [2] considered caging as a prerequisite for a form closure grasp by introducing a notion of a pregrasping cage. Starting from a pregrasping cage, a manipulator can move to a form closure grasp without breaking the cage, hence guaranteeing that the object cannot escape during this process.

One can derive sufficient caging conditions for caging tools of more complex shapes by considering more complex geometric and topological representations. For example, an approach towards caging 3D objects with ‘holes’ was proposed by some of the authors in [12, 13, 14]. Another shape feature was later proposed in [15], where we presented a method to cage objects with narrow parts as seen

¹https://people.kth.se/~mwelle/pc_datasets.html

in Fig. A.2 on the right-hand side. Makita et al. [16, 17] have proposed sufficient conditions for caging objects corresponding to certain geometric primitives.

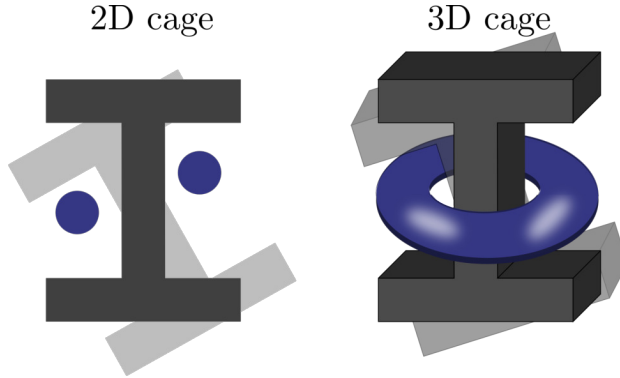


Figure A.2: Example of a 2D cage (left) and a 3D cage exploiting a narrow part of the object.

Finally, research has studied the connectivity of the free space of the object by explicitly approximating it. For instance, Zhang et al. [18] use approximate cell decomposition to check whether pairs of configurations are disconnected in the free space. Another approach was proposed by Wan and Fukui [19], who studied cell-based approximations of the configuration space based on sampling. McCarthy et al. [3] proposed to randomly sample the configuration space and reconstruct its approximation as a simplicial complex. Mahler et al. [6, 7] extend this approach by defining, verifying and generating *energy-bounded cages* – configurations where physical forces and obstacles complement each other in restricting the mobility of the object. These methods work with polygonal objects and caging tools of arbitrary shape, and therefore are applicable to a much broader set of scenarios. However, these approaches are computationally expensive, as discretizing and approximating a three-dimensional configuration space is not an easy task.

To enable a robot to quickly evaluate the quality of a particular configuration and to decide how to place its fingers, we design, train and evaluate a neural network that approximates our caging evaluation function (see [20] for an overview of data-driven grasping). This approach is inspired by recent success in using deep neural networks in grasping applications, where a robot policy to plan grasps is learned on images of target objects by training on large datasets of images, grasps, and success labels. Many experiments suggest that these methods can generalize to a wide variety of objects with no prior knowledge of the object’s exact shape, pose, mass properties, or frictional properties [21, 22, 23]. Labels may be curated from human labelers [24, 25, 26], collected from attempts on a physical robot [27, 28], or generated from analysis of models based on physics and geometry [29, 30, 31, 32]. We explore the latter approach, developing a data-driven partial caging evaluation framework. Our pipeline takes images of an object and caging tools as input and

outputs (i) whether a configuration is a partial cage and (ii) for each partial caging configuration, a real number corresponding to a predicted clearance, which is then used to rank the partial caging configuration.

Generative approaches to training dataset collection for grasping typically fall into one of three categories: methods based on probabilistic mechanical wrench space analysis [32], methods based on dynamic simulation [29, 31], and methods based on geometric heuristics [30]. Our work is related to methods based on grasp analysis, but we derive a partial caging evaluation function based on caging conditions rather than using mechanical wrench space analysis.

3 Partial Caging and Clearance

3.1 Partial Caging

In this section, we discuss the notion of partial caging defined in [8]. Let \mathcal{C} be the configuration space of the object², $\mathcal{C}_{col} \subset \mathcal{C}$ be its subset containing configurations in collision, and let $\mathcal{C}_{free} = \mathcal{C} - \mathcal{C}_{col}$ be the free space of the object. Let us assume \mathcal{C}_{col} is bounded. Recall the traditional definition of caging:

Definition 1. *A configuration $c \in \mathcal{C}_{free}$ is a cage if it is located in a bounded connected component of \mathcal{C}_{free} .*

In practical applications, it may be beneficial to identify not just cages, but also configurations which are in some sense ‘close’ to a cage, i.e., configurations from which it is difficult but not necessarily impossible to escape. Such partial caging can be formulated in a number of ways: for example, one could assume that an object is partially caged if its mobility is bounded by physical forces, or it is almost fully surrounded by collision space but still can escape through narrow openings.

We introduce the maximal clearance of an escaping path as a quality measure. Intuitively, we are interested in partial caging configurations where an object can move within a connected component, but can only escape from it through a narrow passage. The ‘width’ of this narrow passage then determines the quality of a configuration.

Let us now provide the necessary definitions. Since, by our assumption, the collision space of the object is bounded, there exists a ball $B_R \subset \mathcal{C}$ of a finite radius containing it. Let us define the escape region $X_{esc} \subset \mathcal{C}$ as the complement of this ball: $X_{esc} = \mathcal{C} - B_R$.

Definition 2. *A collision-free path $p : [0, 1] \rightarrow \mathcal{C}_{free}$ from a configuration c to X_{esc} is called an escaping path. The set of all possible escaping paths is denoted by $\mathcal{EP}(\mathcal{C}_{free}, c)$.*

²Note that in this paper we focus on the case where $\mathcal{C} \subset SE(2)$, but the definition of partial caging holds for arbitrary configuration spaces

Let $cl : \mathcal{EP}(\mathcal{C}_{free}, c) \rightarrow \mathbb{R}_+$ be a cost function defined as the minimum distance from the object along the path p to the caging tools: $cl(p) = \min_{c \in p} (\text{dist}(o_c, \mathbf{g}))$ where o_c is the object placed in the configuration c and \mathbf{g} denotes the caging tools. We define the caging evaluation function as follows:

$$Q_{cl}(c) = \begin{cases} \min_{p \in \mathcal{EP}(\mathcal{C}_{free}, c)} cl(p), & \mathcal{EP}(\mathcal{C}_{free}, c) \neq \emptyset \\ 0, & \mathcal{EP}(\mathcal{C}_{free}, c) = \emptyset. \end{cases}$$

3.2 The set \mathcal{C}_{cage}

Observe that a low value of clearance measure on arbitrary configurations of \mathcal{C}_{free} does not guarantee that a configuration is a sufficiently “good” partial cage. For example, consider only one convex caging tool located close to the object as in Fig. A.3 (left). In this case, the object can easily escape. However, the clearance of this escaping path will be low, because the object is initially located very close to the caging tool. The same clearance value can be achieved in a much better partial caging configuration, see Fig. A.3 (right). Here, the object is almost completely surrounded by a caging tool, and it can escape through a narrow gate. Clearly, the second situation is much preferable from the caging point of view. Therefore, we would like to be able to distinguish between these two scenarios.

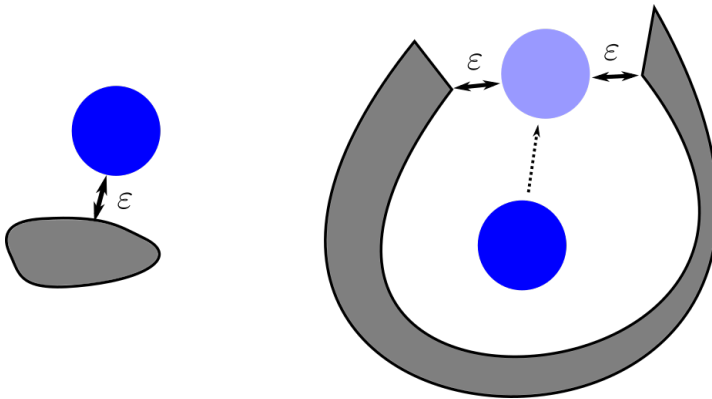


Figure A.3: On the left, an object (blue) can easily escape from the caging tool (grey); on the right, the object is partially surrounded by the caging tool and escaping is therefore harder. Both escaping paths will have the same clearance ε .

Assume that caging tools are placed such that the object can escape. We increase the size of the caging tools by an offset, and eventually, for a sufficiently large offset, the object collides with the enlarged caging tools; let us assume that the size of the offset at this moment is $\varepsilon_{col} > 0$. We are interested in those configurations for which there exists an intermediate size of the offset $0 < \varepsilon_{closed} < \varepsilon_{col}$, such that the object is caged by the enlarged caging tools, but is not in collision. This is not

always possible, as in certain situations the object may never become caged before colliding with enlarged caging tools. Fig. A.4 illustrates this situation.

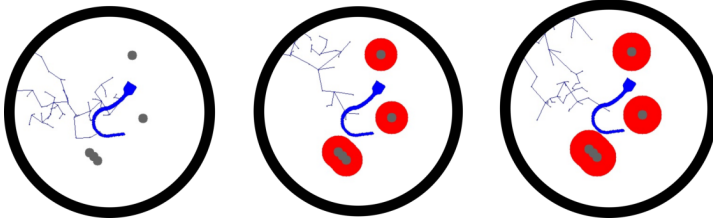


Figure A.4: The object (hook) is shown in blue while the caging tools are gray, the red symbolises the enlargement of the caging tools. The RRT nodes and edges are depicted in purple. From left to right, three enlargements of the caging tools are depicted. The object can always escape until its initial configuration stops being collision-free.

Let us formally describe this situation. Let $\mathcal{C}_{free}^\varepsilon$ be the free space of the object induced by ε -offset of caging tools. As we increase the size of the offset, we get a nested family of spaces $\mathcal{C}_{free}^{\varepsilon_{col}} \subset \dots \subset \mathcal{C}_{free}^\varepsilon \subset \dots \subset \mathcal{C}_{free}^0$, where ε_{col} is the smallest size of the offset causing a collision between the object and the enlarged caging tools. There are two possible scenarios: in the first one, there is a value $0 < \varepsilon_{closed} < \varepsilon_{col}$ such that when the offset size reaches it the object is caged by the enlarged caging tools. This situation is favorable for robotic manipulation settings, as in this case the object has some freedom to move within a partial cage, but cannot escape arbitrarily far as its mobility is limited by a narrow gate (see Fig. A.5)³.

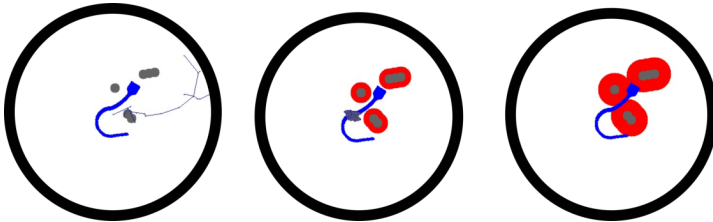


Figure A.5: From left to right: the object (hook) can escape only in the first case, and becomes completely caged when we enlarge the caging tools before a collision with the object occurs.

We denote the set of all configurations falling into this category as the *caging subset* \mathcal{C}_{cage} . These configurations are promising partial cage candidates, and our primary interest is to identify these configurations. In the second scenario, for any ε between 0 and ε_{col} , the object is not caged in the respective free space $\mathcal{C}_{free}^\varepsilon$, as shown in Fig. A.4.

³In Fig. A.5 the enlarged caging tools (in red) cage the hook by trapping the larger base.

We define the notion of *partial caging* as follows:

Definition 3. Any configuration $c \in \mathcal{C}_{cage}$ of the object is called a *partial cage of clearance* $Q_{cl}(c)$.

Note that the case where $\mathcal{EP}(\mathcal{C}_{cage}, c) = \emptyset$ corresponds to the case of a complete (i.e., classical) cage. Thus, partial caging is a generalization of complete caging.

Based on this theoretical framework, we propose a partial caging evaluation process that consists of two stages. First, we determine whether a given configuration belongs to the caging subset \mathcal{C}_{cage} . If it does, we further evaluate its clearance with respect to our clearance measure Q_{cl} , where, intuitively, configurations with smaller clearance are considered more preferable for grasping and manipulation under uncertainty.

4 Gate-Based Clearance Estimation Algorithm

Algorithm 1 Gate-Based Clearance Estimation

Require: object O , caging tools G , ε_{max}

```

 $\varepsilon_{min} \leftarrow 0$ 
while Can-Escape( $O$ ,  $G$ ,  $\varepsilon_0$ ) do
     $\varepsilon_{cl} \leftarrow (\varepsilon_{min} + \varepsilon_{max})/2$ 
    if Can-Escape( $O$ ,  $G$ ,  $\varepsilon_{cl}$ ) then
         $\varepsilon_{min} \leftarrow \varepsilon_{cl}$ 
    else
         $\varepsilon_{max} \leftarrow \varepsilon_{cl}$ 
return  $\varepsilon_{cl}$  # clearance of an escaping path

```

In this section, we propose a possible approach to estimate $Q_{cl}(c)$ – the Gate-Based Clearance Estimation Algorithm. Instead of finding a path with maximum clearance directly, we gradually inflate the caging tools by a distance offset until the object becomes completely caged. For this, we first approximate the object and the caging tools as union of discs, see Fig. A.8. This makes enlarging the caging tools an easy task – we simply increase the radii of the discs in the caging tools’ approximation by a given value. The procedure described in Alg. 1 is then used to estimate $Q_{cl}(c)$.

We perform bisection search to find the offset value at which an object becomes completely caged. For this, we consider offset values between 0 and the radii of the workspace. We run RRT at every iteration of the bisection search in order to check whether a given value of the offset makes the object caged. In the experiments, we choose a threshold of 4 million iterations⁴ and assume that the object is fully

⁴Our experimental evaluation for our test dataset suggested that if after 4 million iterations RRT had not found an escaping path, then the object was caged with overwhelming likelihood. We thus considered RRT with this setting to provide a sufficiently good approximation for training the neural network.

caged if RRT does not produce an escaping path at this offset value. Note that this procedure, due to the approximation with RRT up to a maximal number of iterations, does not guarantee that an object is fully caged; however, since no rigorous bound on the number of iterations made by RRT is known, we choose a threshold that performs well in practice since errors due to this RRT-based approximation become insignificant for sufficiently large maximal numbers of RRT sampling iterations. In Alg. 1, $\text{Can-Escape}(O, G, \varepsilon_{cl})$ returns *True* if the object can escape and is in a collision-free configuration.

5 Grasping favorable configuration in \mathcal{C}_{cage}

Depending on the size of the object with respect to the workspace, the bisection search performed in Alg. 1 can be computationally expensive. Uniformly sampling caging tools placements from the entire workspace in order to find configurations in \mathcal{C}_{cage} is also rather inefficient and the number of partial caging configurations of high quality can be low.

Furthermore, not all partial caging configurations defined by Def. 3 ($c \in \mathcal{C}_{cage}$) are equally suitable for certain applications like grasping or pushing under uncertainty. Namely, we would like to place caging tools such that they are not too close and not too far away from the object.

To overcome these limitations, we define a region around the object called *partial caging grasping band* (Fig. A.6 illustrates this concept):

Definition 4. Let O be an object and assume the caging tools have a maximal width⁵ ct_d . Let O_{min} and O_{max} be objects where the composing disks are enlarged by $dis_{min} = \frac{1}{2}ct_d \cdot (1 + \beta)$ and $dis_{max} = dis_{min} + \frac{1}{2}ct_d \cdot \gamma$ respectively.

We can then define the grasping band as follows:

$$\mathcal{GB} = \{x \in \mathcal{C}_{free} : (x \in O_{min}) \oplus (x \in O_{max})\},$$

Here, β and γ are parameters that capture the impreciseness of the system, such as vision and control uncertainties.

6 Learning Planar Q_{cl}

As RRT is a non-deterministic algorithm, one would need to perform multiple runs in order to estimate Q_{cl} . In real-time applications, we would like the robot to be able to evaluate caging configurations within milliseconds. Thus, the main obstacle on the way towards using the partial caging evaluation function defined above in real time is the computation time needed to evaluate a single partial caging configuration.

⁵The caging tools are composed of disks with ct_d as diameter. As we only consider composed line configurations as caging tools the width never exceeds ct_d .

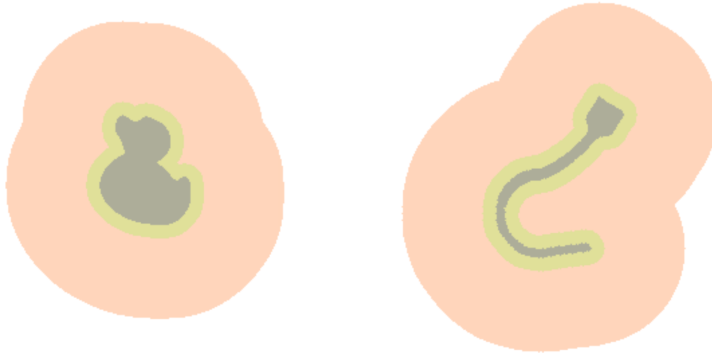


Figure A.6: An illustration of a grasping band for a duck and hook object. The object O is in the center (gray) overlaid by O_{min} (O enlarged by dis_{min} , light green) overlaid by O_{max} (O enlarged by dis_{max} , light orange). The grasping band (\mathcal{GB}) is the disjunctive union between O_{min} and O_{max} .

Alg. 1 requires several minutes to evaluate a single partial cage, while a neural network can potentially estimate a configuration in less than a second.

To address this limitation of Alg. 1, we design and train two convolutional neural networks. The first, called *CageMaskNN*, acts as a binary classifier that identifies configurations that belong to \mathcal{C}_{cage} following Def 3. The second, architecturally identical network, called *CageClearanceNN*, approximates the caging evaluation function Q_{cl} to estimate the quality of configurations. The network takes two images as input that correspond to the object and the caging tools. The two networks are separated to make training more efficient, as both can be trained independently. Operating both networks sequentially results in pipeline visualized in Fig. A.1: first, we identify if a configuration is a partial cage, and if it is, we evaluate its quality.

Our goal is to estimate Q_{cl} given $O \subset \mathbb{R}^2$ – an object in a fixed position, and $G = \{g_1, g_2, \dots, g_n\}$ – a set of caging tools in a particular configuration. We assume that caging tools are normally disconnected, while objects always have a single connected component. In our current implementation, we consider $n \in \{3, 4\}$, and multiple caging tool shapes.

While neural networks require a significant time to train (often multiple hours), evaluation of a single configuration is a simple forward pass through the network and its complexity is therefore not relying on the input size or data size but rather on the number of neurons in the network. In this work, our goal is to show that we can successfully train a neural network that can generalise to unseen input configurations and approximate the algorithm 1 in milliseconds.

6.1 Dataset Generation

We create a dataset of 3811 object models consisting of two-dimensional slices of objects’ three-dimensional mesh representations created for the Dex-Net 2.0 framework [32]. We further approximate each model as a union of one hundred discs, to strike a balance between accuracy and computational speed. The approximation error is a ratio that captures how well the approximation (A_{app}) represents the original object (A_{org}), and is calculated as follows: $a_e = \frac{A_{org} - A_{app}}{A_{org}}$. Given the set of objects, two partial caging datasets are generated. The first dataset, called *PC-general*, consists of 3811 objects, 124435 partial caging configurations (belonging to \mathcal{C}_{cage}), and 18935565 configurations that do not belong to \mathcal{C}_{cage} .

One of the limitations of the *PC-general* dataset is that it contains relatively few partial caging configurations of high quality. To address this limitation, generate a second partial caging dataset called *PC-band* where caging tools placements are only located inside the grasping bands of objects, as this strategy increases the chance that the configuration will be a partial cage of low Q_{cl} as well as the likelihood of a configuration belonging to \mathcal{C}_{cage} .

The *PC-band* dataset consists of 772 object with 3,785,591 configurations of caging tools, 127,733 of which do belong to the partial caging subset \mathcal{C}_{cage} . We set β to the approximation error a_e for each object and $\gamma = 6$ to define the grasping band.

All configurations are evaluated with Q_{cl} (see algorithm 1). The distribution of partial cages can be seen in Fig. A.7.

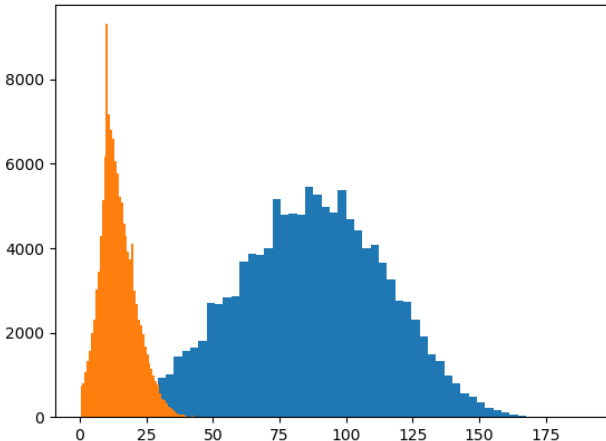


Figure A.7: Distribution of Q_{cl} estimates for the *PC-general* (blue) and the *PC-band*(orange) datasets.

Examples of configurations for both datasets can be seen in Fig. A.8. The disk approximation of the object is shown in blue, while the original object is depicted in red. PC -general contains configurations placed in the entire workspace while PC -band is limited to configuration sampled inside the grasping band.

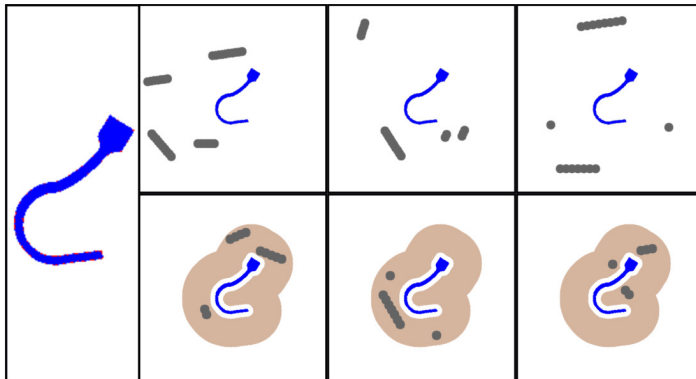


Figure A.8: Left: original representations of a hook objects (red) and in blue their approximation by a union of discs of various sizes closely matching the polygonal shape ($a_e = 0.051$); second and third column: configurations that do not belong to \mathcal{C}_{cage} ; last column: a partial caging configuration ($c \in \mathcal{C}_{cage}$). The top row is from PC -general, the bottom from PC -band.

6.2 Architecture of Convolutional Neural Networks

We propose a multi-resolution architecture that takes the input image as $64 \times 64 \times 2$, $32 \times 32 \times 2$, and $16 \times 16 \times 2$ tensors. This architecture is inspired by inception blocks [33]. The idea is that the global geometric structure can be best captured with different image sizes, such that the three different branches can handle scale-sensitive features. The network $CageMaskNN$ determines whether a certain configuration belongs to \mathcal{C}_{cage} , while $CageClearanceNN$ predicts the clearance Q_{cl} value for a given input configuration.

The architecture of the networks is shown in Fig. A.9. Both networks take an image of an object and caging tools on a uniform background position and orientation belonging to the same coordinate frame constituting a two-channel image ($64 \times 64 \times 2$) as input. $CageMaskNN$ performs binary classification of configurations by returning 0 in case a configuration belongs to \mathcal{C}_{cage} , and 1 otherwise. $CageClearanceNN$ uses clearance Q_{cl} values as labels and outputs a real value – the predicted clearance of a partial cage. The networks are trained using the Tensorflow [34] implementation of the Adam algorithm [35]. The loss is defined as the mean-squared-error (MSE) between the prediction and the true label. The batch size was chosen to be 100 in order to compromise between learning speed

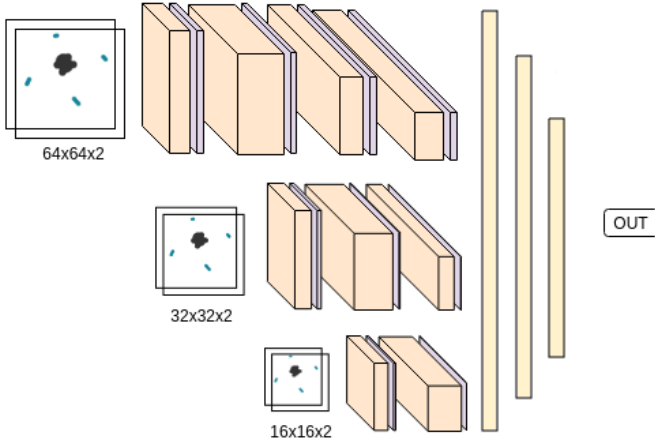


Figure A.9: As caging depends on global geometric properties of objects, a CNN architecture with multi-resolution input was designed to capture these features efficiently.

and gradient decent accuracy. The networks were trained on both of our datasets – *PC-general* and *PC-band*.

7 Training and evaluation of the networks

In this section we describe how we train and evaluate the two networks and perform an ablation study of the architecture. In detail, for *CageMaskNN*, we investigate to what extent the training data should consist of samples belonging to \mathcal{C}_{cage} and evaluate the performance of the best such composition against a simpler network architecture. Following that, we investigate how the number of different objects as well as the choice of dataset influences the performance of *CageMaskNN*.

For *CageClearanceNN*, we also perform an analysis of the effect of the the number of objects in the training data and to what extent the choice of dataset influences the performance and compare it to a simpler architecture. As a final investigation, we investigate the error for specific Q_{cl} intervals.

Note that the training data is composed of samples where the ground truth of the configuration was obtained using algorithm 1. A main goal of the presented evaluation is hence to investigate how well the proposed networks are able to generalise to examples that were not included in the training data (unseen test data). High such generalization performance, is a key indicator for the potential application of the proposed fast neural network based approach (execution in milliseconds) instead of the computationally expensive underlying algorithm 1 (execution in minutes) that was used to generate the training data.

Single-res Architecture: In order to perform an ablation of the previous discussed multi-resolution architecture we compare the performance so a architecture that has only a single resolution as input. The *Single-res Arch.* Takes only the 64x64x2 as input and is missing the other heads completely. In this way we want to see if our assumption that different sized inputs are beneficial to the networks performance.

7.1 CageMaskNN - % of C_{cage} and Ablation

We generate 4 datasets containing 5%, 10%, 15%, and 20% caging configurations in C_{cage} respectively from *PC-general*. This is achieved by oversampling as well as by performing rotational augmentation with 90, 180 and 270 degrees of the existing caging configurations. The *Single-res Arch.* is trained with 10% caging configurations in C_{cage} for comparison.

The evaluation is performed on a test set consisting of 50% caging examples from C_{cage} . In Fig. A.10, we show the F1-curve and Accuracy-curve. All five versions of the network where trained with 3048 objects with 2000 configuration each, using a batch size of 100 and 250000 iterations. To avoid overfitting, a validation set of 381 objects is evaluated after every 100th iteration. The final scoring is done on a test set consisting of 381 previously unseen objects. The mean squared error (MSE) on the unseen test set was 0.0758, 0.0634, 0.0973 and 0.072 for the 5%, 10%, 15% and 20% version respectively, indicating that *CageMaskNN* is able to generalize to novel objects and configurations from our test set. The MSE for the single resolution network was 0.155 showing the significant gain obtained by utilizing the multi-resolution branches.

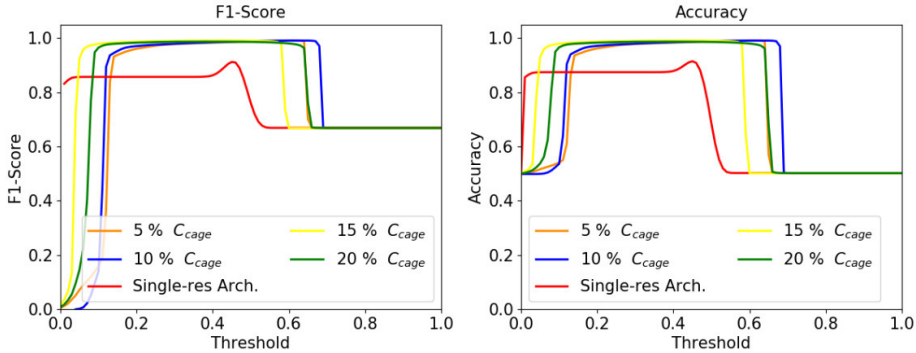


Figure A.10: F1-score and accuracy of the network depending on different thresholds

We observe that the network that was trained on the dataset where 10% of the configurations are partial cages performs slightly better than the other versions. Note however that only the one that was trained with 5% of partial cages performs

significantly worse. All versions of the multi-resolution architecture outperform the *Single-res Arch*, which justifies our architecture design.

7.2 CageMaskNN - Number of Objects and Datasets

We investigate how the performance of the networks depends on the size of the training data and how the two training datasets, *PC-general* and *PC-band*, affect the performance of the networks. Table A.1 shows the area under ROC curve (AUC) and the average precision (AP) for *CageMaskNN* for training set composed of 1, 10, 100, and 1000 objects from the dataset *PC-general*, as well as 1, 10, 100, and 617 objects from *PC-band*. We observe that having more objects in the training set results in better performance. We note that the network trained on *PC-general* slightly outperforms the one trained on *PC-band*.

Training set	pc-general		pc-band	
	AUC	AP	AUC	AP
1 object	0.92	0.88	0.88	0.83
10 objects	0.91	0.88	0.88	0.84
100 objects	0.97	0.92	0.92	0.89
1000 617 objects	1.00	1.00	1.00	0.96

Table A.1: The area under ROC curve (AUC) and the average precision (AP) for different training set constitutions, evaluated on the test set with 50 % of partial cage configurations. In all training sets 10 % of configurations belong to C_{cage} . We observe that *PC-general* has a slightly better performance than *PC-band*.

Fig. A.11 demonstrates how the performance of the networks increases with the number of objects in the training dataset by showing the F1-score as well as the accuracy for both datasets. We observe that the network, independently of the training dataset, demonstrates acceptable performance even with a modest numbers of objects in the training dataset. One key factor here is the validation set which decreases the generalisation error by choosing the best performance during the entire training run, thus reducing the risk of overfitting. Similarly to the previous results, *PC-general* slightly outperforms *PC-band*.

7.3 CageClearanceNN - Number of Objects and Ablation

The purpose of *CageClearanceNN* is to predict the value of the clearance measure Q_{cl} given a partial caging configuration. We trained *CageClearanceNN* on 1, 10, 100, 1000 and 3048 objects from *PC-general* as well as a single resolution variant with the same training sets. Additionally, we trained another instance of *CageClearanceNN* with 1, 10, 100, and 617 objects from *PC-band*, and the corresponding single-resolution architecture version for each number of objects. The

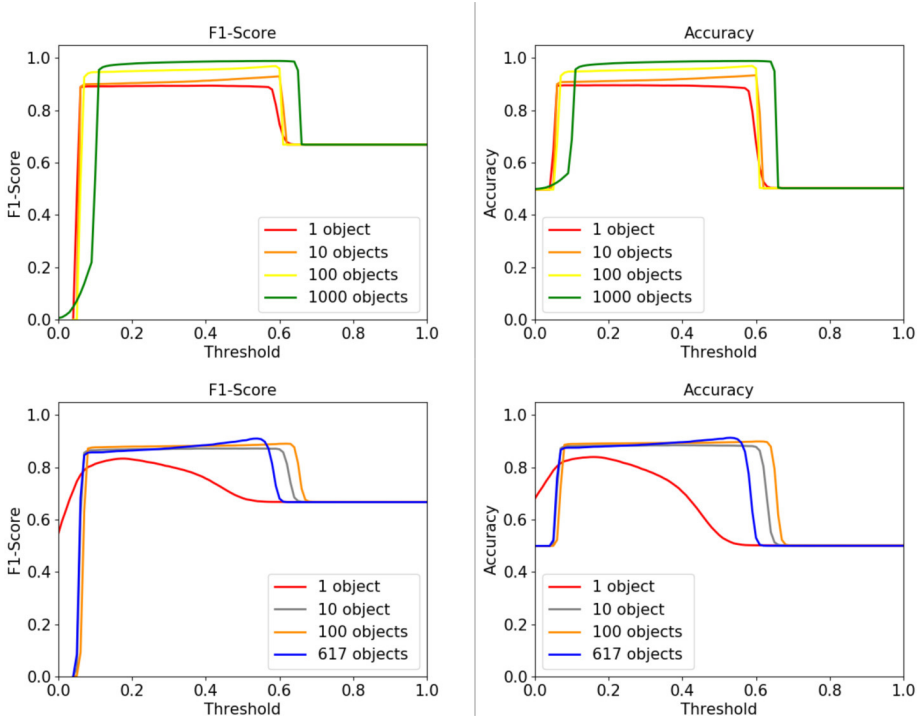


Figure A.11: F1-score and accuracy of the network trained with 1, 10, 100, and 1000 || 617 objects, for *PC-general* (top row) and *PC-band* (bottom row) respectively on a test set with 50 % C_{cage} configuration .

label is scaled with a factor of 0.1, as we found that the networks performance improves for smaller training input values. The left-hand side of Fig.A.12 shows a rapid decrease of MSE as we increase the number of training data objects to 1000, and a slight performance increase between 1000 and 3048 training objects for the *PC-general* dataset. We can also see that employing the multi-resolution architecture only leads to significant performance increase when going up to 1000 objects and more. The right-hand side of Fig.A.12 presents the analogous plot for the network trained on *PC-band*. We observe the same rapid decrease of MSE as we include more objects in the training set. Note that the different number of parameter plays a role as well in the performance difference. Since our current dataset is limited to 617 training examples of object shapes, we do not observe the benefits of the multi-resolution architecture. Note that the difference in absolute MSE stems from the different distributions of the two datasets (as can be seen in Fig. A.7). This indicates that further increases in performance can be gained by having more training objects. Increasing the performance for more than 3000 objects may however require a significant upscaling of the training dataset.

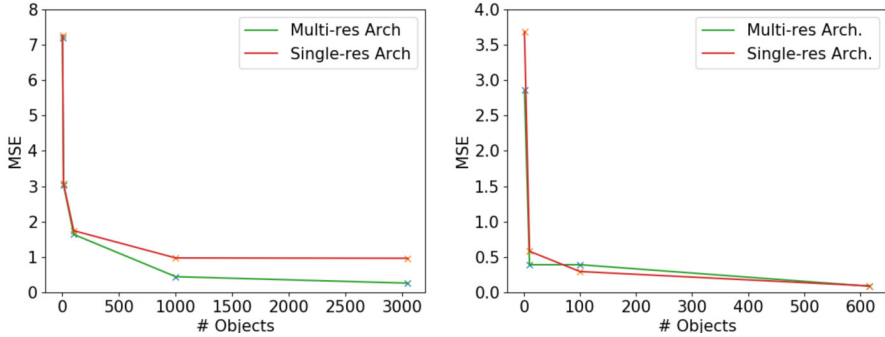


Figure A.12: left: MSE of *CageClearanceNN* trained on *PC-general* with different numbers of objects and a single-resolution architecture; right: MSE of the single-resolution architecture trained on *PC-band* with different numbers of objects.

7.4 CageClearanceNN - Error for specific Q_{cl}

We investigated the MSE for specific Q_{cl} value intervals. Fig. A.13 shows the MSE on the test set with respect to the Q_{cl} values (as before, scaled by 0.1). Unsurprisingly, we observe that the network, trained on *PC-general*, that was trained only on one object, does not generalise over the entire clearance/label spectrum. As we increase the number of objects, the performance of the network increases. The number of outliers with large errors decreases significantly when the network is trained on 1000 objects. On the right side, we can see the MSE for the final *CageClearanceNN* network trained on *PC-general*. We observe that low values of Q_{cl} are associated to higher error values. Analysing this behavior on *CageClearanceNN* trained on *PC-band* demonstrates a very similar behavior and is therefore omitted.

8 Planar Caging Pipeline Evaluation

8.1 Last caging tool placement

In this experiment, we consider the scenario where $n - 1$ out of n caging tools are already placed in fixed locations, and our framework is used to evaluate a set of possible placements for the last tool to acquire a partial cage. We represent possible placements as cells of a two-dimensional grid and assume that the orientation of the caging tool is fixed. Fig. A.15 illustrates this approach. We use the pipeline trained with *PC-general* as it covers the entire workspace. In the example *a*, we can see that placing the caging tool closer to the object results in better partial caging configurations. This result is consistent with our definition of the partial caging quality measure. We note furthermore, that *CageMaskNN* obtains an approximately cor-

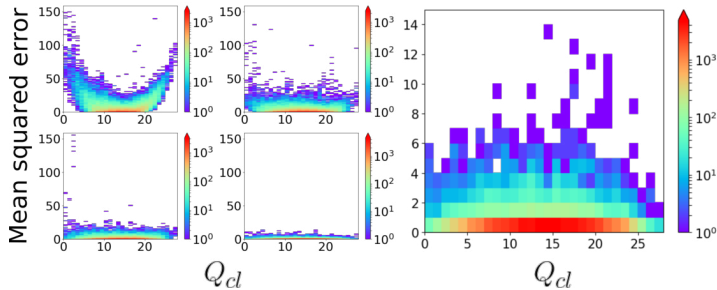


Figure A.13: MSE for each test case sorted for labels. Left: shows performance of 1, 10, 100, 1000 objects (top left, top right, bottom left, bottom right). Right: shows MSE of entire test set for the final *CageClearanceNN*. Note that the figure on the right is zoomed in as errors are significantly smaller (see the left y-axis of that figure).

rect region-mask of partial caging configurations for this novel object. Example *b* demonstrates the same object with elongated caging tools. Observe that this results in a larger region for possible placement of the additional tool. Example *c* depicts the same object but the fixed disc-shaped caging tool has been removed and we are considering three instead of four total caging tools. This decreases the number of possible successful placements for the additional caging tool. We can see that our framework determines the successful region correctly, but is more conservative than the ground truth. In the example *d*, we consider an object with two large concavities and three caging tools. We observe that *CageMaskNN* identifies the region for C_{cage} correctly and preserves its connectivity. Similarly to the previous experiments, we can also observe that the most promising placements (in blue) are located closer to the object.

8.2 Evaluating Q_{cl} along a trajectory

We now consider a use case of Q_{cl} along a caging tool trajectory during manipulation enabled by the fact that the evaluation of a single caging configuration using *CageMaskNN* and *CageClearanceNN* takes less than 6ms on a GeForce GTX 1080 GPU.

The results for two simulated sample trajectories are depicted in Fig. A.14. In the first row, we consider a trajectory of two parallel caging tools, while in the trajectory displayed in the bottom row, we consider the movement of 4 caging tools: caging tool 1 moves from the top left diagonally downwards and then straight up, caging tool 2 enters from the bottom left and then exits towards top, caging tool 3 enters from the top right and then moves downwards, while caging tool 4 enters from the bottom right and then moves downwards.

The identification of partial caging configurations by *CageMaskNN* is rather

stable as we move the caging tool along the reference trajectories, but occurs at a slight offset from the ground truth. The offset in *CageClearanceNN* is larger but consistent, which can be explained by the fact that similar objects seen during training had a lower clearance as the novel hourglass shaped object. In the second example, the clearance of the partial cage decreases continuously as the caging tools get closer to the object. Predicted clearance values from *CageClearanceNN* display little noise and low absolute error relative to the ground truth. Note that a value of -1 in the quality plots refers to configurations identified as not being in \mathcal{C}_{cage} by *CageMaskNN*.

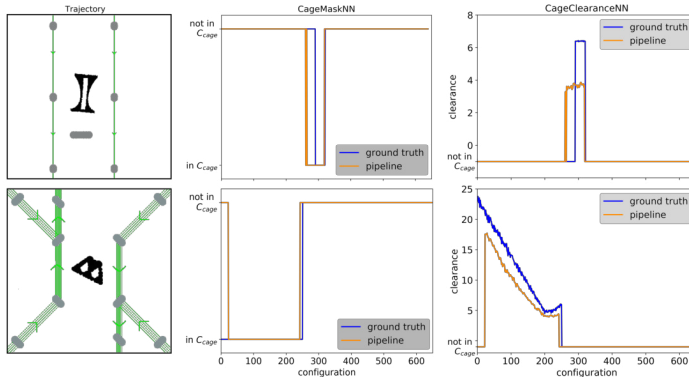


Figure A.14: Evaluation of the pipeline along two trajectories. The trajectory (left, green) is evaluated with *CageMaskNN* (middle) and *CageClearanceNN* (right), which evaluates Q_{cl} for those configurations where *CageMaskNN* returns 0. The predictions by the networks are displayed in orange while ground truth is shown in blue.

8.3 Experimental evaluation of Q_{cl}

In this section, we experimentally evaluate our partial caging quality measure Q_{cl} by simulating random shaking of the caging tools and measuring the needed time for the object to escape. Intuitively, the escape time should be inversely proportional to the estimated Q_{cl} ; this would indicate that it is difficult to escape the partial cage. A similar approach to partial caging evaluation has been proposed in [5]. Where the escape time was computed using probabilistic motion planning methods like RRT, RRT*, PRM, SBL as well as a random planner was measured.

8.3.1 Random partial caging trajectories

We apply a simple random walk X_n as a sequence of independent random variables S_1, S_2, \dots, S_n where each S is randomly chosen from the set $\{(1, 0), (0, 1), (1, 1), (-1, 0), (0, -1), (-1, -1)\}$ with equal probability.

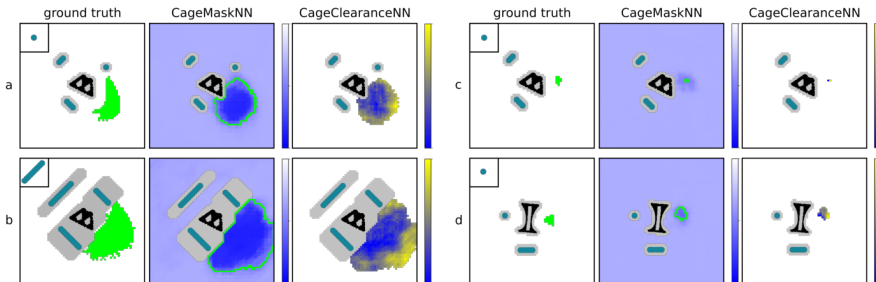


Figure A.15: Here, we depict the results of four different experiments. The green region indicates configuration where the additional caging tool completes the configuration in such a way that the resulting configuration is a partial cage. The small squares in the ground truth figures depict the caging tools that are being placed (for simplicity the orientations are fixed). We plot the output for each configuration directly and visualize the result as a heatmap diagram (blue for partial caging configurations, white otherwise). The best placements according to *CageClearanceNN* are depicted in dark blue, and the worst ones in yellow. The results are normalized between 0 and 1. Grey area corresponds to the placements that would result in a collision.

$$X_n = X_0 + S_1 + S_2 + \dots + S_n),$$

where X_0 is the start position of the caging tools. and a stride factor α determines at what time the next step of the random walk is performed.

In this experiment, unlike in the rest of the paper, caging tools are moving along randomly generated trajectories. We assume that the object escapes a partial cage when it is located outside of the convex hull of the caging tools. If the object does not escape within t_{max} seconds, the simulation is stopped. The simulation is performed with the software pymunk that is build on the physic engine Chipmunk 2D [36]. We set the stride factor $\alpha = 0.05s$ so that a random step S of the random walk X_n is applied to the caging tool every 0.05 seconds. As pymunk also facilitates object interactions, the caging tool can push the object around as well as drag it with them. Figure A.16 illustrates this process.

The experiment was performed on 5 different objects, depending on the object we used between 437-1311 caging tool configurations. For each of them the escape time was estimated as described above. As it is not deterministic, we performed 100 trials for each configuration and computed the mean value. The mean escape time of 100 trials was normalized such that the values range between 0 and 1. Furthermore, for each configuration we computed Q_{cl} and the Pearson correlation coefficient⁶. Fig. A.17 illustrates the results.

⁶The Pearson correlation coefficient measures the linear correlation between the escape time from random shaking and the defined clearance measure Q_{cl} .

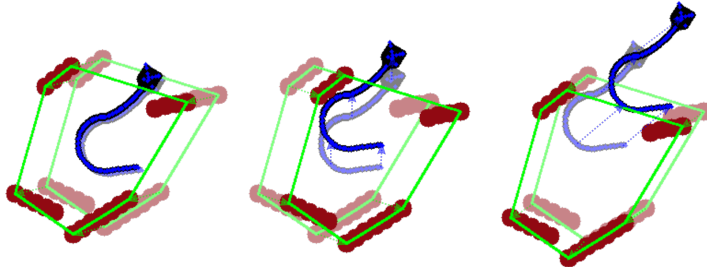


Figure A.16: Random trajectory for caging tools. Left: time $t = 0s$ (transparent) to $t = 0.83s$ (not escaped), middle: $t = 0.83s$ (transparent) to $t = 1.67s$ (not escaped), right: time $t = 1.67s$ (transparent) to $t = 2.47s$ (escaped). Note that the caging tools do not necessarily run in a straight line but rather follow the randomly generated trajectory with a new step every $0.05s$. As a simple physics simulator is used, the caging tools can also induce movement of the object by colliding with it.

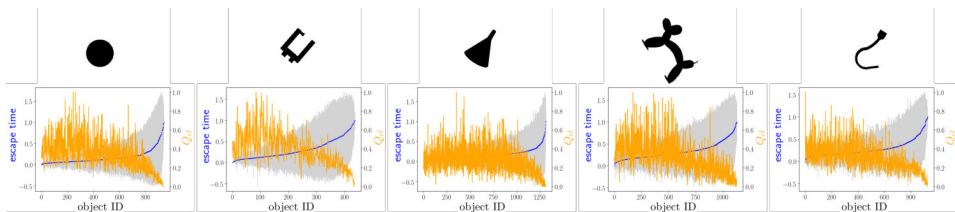


Figure A.17: Correlation between escape time from random shaking and Q_{cl} . Top row shows evaluated objects (disk, clench, cone, balloon animal, and hook, on the bottom row the partial cages are sorted according to respective average escape time, and plot the average escape time (in blue), its variance (in gray), and Q_{cl} (in orange). Pearson correlation coefficient of the escape time and Q_{cl} (from left to right) are: -0.608 , -0.462 , -0.666 , -0.566 , -0.599

Our results show that the longer it takes for the object to escape the partial cage, the higher the variance of the escape time is. This indicates that a partial cage quality estimate based on the average escape time would require a high number of trials, making the method inefficient.

Furthermore, we demonstrate that our clearance-based partial caging quality measure shows a trend with the average escape time for strong partial cages, which suggests the usefulness of the proposed measure.

8.4 Different metrics in the space of shapes for partial caging

A natural extension of our partial caging evaluation framework is partial cage acquisition: given a previously unseen object, we would like to be able to quickly

synthesise partial cages of sufficient quality. In this section, we make the first step in this direction, and propose the following procedure: given a novel object, we find similar objects from the training set of the *PC-band*, and consider those partial caging configurations that worked well for these similar objects.

The key question here is how to define a distance function for the space of objects that would capture the most relevant shape features for partial caging. In this experiment, we investigate three different shape distance functions: Hausdorff distance, Hamming distance, and Euclidean distance in the latent space of a variational autoencoder, trained on the set of objects used in this work. Variational autoencoders (VAEs) are able to encode high-dimensional input data into a lower-dimensional latent space while training in an unsupervised manner. In contrast to a standard encoder/decoder setup, which returns a single point, a variational autoencoder returns a distribution over the latent space, using the *KL*-cost term as regularisation.

We evaluate different distance functions with respect to the quality of the resulting partial cages. Given a novel object, we calculate the distance to each known object in the dataset according to the three distance functions under consideration, and for each of them we select five closest objects. When comparing the objects, orientation is an important factor. We compare 360 rotated versions of the novel object with the known objects from the dataset and pick the one closest following the chosen metric.

8.4.1 VAE-based representation

For our experiment, we train a VAE based on the ResNet architecture with skip connections with six blocks [37] for the encoder and the decoder. The input images have resolution 256x256. We use a latent space with 128 dimensions, dropout of 0.2 and a fully connected layer of 1024 nodes. The VAE loss was defined as follows:

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

The first term achieves reconstruction, while the second term tries to disentangle the distinct features. z denotes latent variable, $p(z)$ the prior distribution, and $q(z|x)$ the approximate posterior distribution. Note that the Bernoulli distribution was used for $p(x|z)$, as the images are of a binary nature. The batch size was set to 32. As the sizes of the objects vary significantly, we invert half of the images randomly when loading a batch. This prevents the collapse to either pure black or pure white images.

8.4.2 Hausdorff distance

The Hausdorff distance is a well known measure for the distance between two sets of points in a metric space (\mathbb{R}^2 for our case). As the objects are represented with disks we use the set of x and y points to represent the object. This is a simplification

of the object as the radius of the circles is not considered. The general Hausdorff distance can be computed with [38]:

$$d_H(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\}$$

8.4.3 Hamming Distance

The Hamming distance [39] is defined as the difference of two binary data strings calculated using the XOR operation. It captures the exact difference between the two images we want to match, as it calculates how many pixel are different. We pre-process the images by subtracting the mean and reshaping the images to a 1D string.

8.5 Performance

We compare the performance of the three different similarity measures, as well as a random selection baseline, on 500 novel object. The percentage of collision-free caging tools placements, as well as the average clearance score is shown in Table A.2. We report the average percentage of collision-free caging tool placements taken from the *PC-band* of partial cages for top 1 and top 5 closest objects.

Furthermore, we evaluate the collision-free configurations using Alg. 1 to provide Q_{cl} values as well as check if the configuration still belongs to \mathcal{C}_{cage} . In the Table A.2, the top 1 column under cage evaluation shows the percentage of configurations that belong to \mathcal{C}_{cage} . To the right is the average Q_{cl} for the most promising cage from the closest object. The top 25 column shows the same results for the five most promising cages for each of the five closest objects. Examples for three novel objects and the closest retrieved objects are shown in Fig. A.18. In the left column, the closest objects with respect to the chosen metric are shown given the novel query object. The right column shows the acquired cages, transferred from the closest known objects. Note that a collision free configuration does not necessarily have to belong to \mathcal{C}_{cage} .

	collision-free		cage evaluation			
			top 1		top 25	
	top 1	top 5	$\in \mathcal{C}_{cage}$	Q_{cl}	$\in \mathcal{C}_{cage}$	Q_{cl}
VAE	90.9%	86.6%	53.2%	5.05	48.4%	5.92
Hausdorff	75.5%	75.2%	33.6%	5.21	32.2%	5.83
Hamming	74.4%	73.9%	35.4%	3.87	34.3%	4.51
Random	61.6%	62.3%	27.0%	13.31	25.4%	14.12

Table A.2: Average results for 500 novel objects cage acquisition using different distance metrics to find similar objects in *PC-band*, and applied cages from retrieved objects to novel objects.

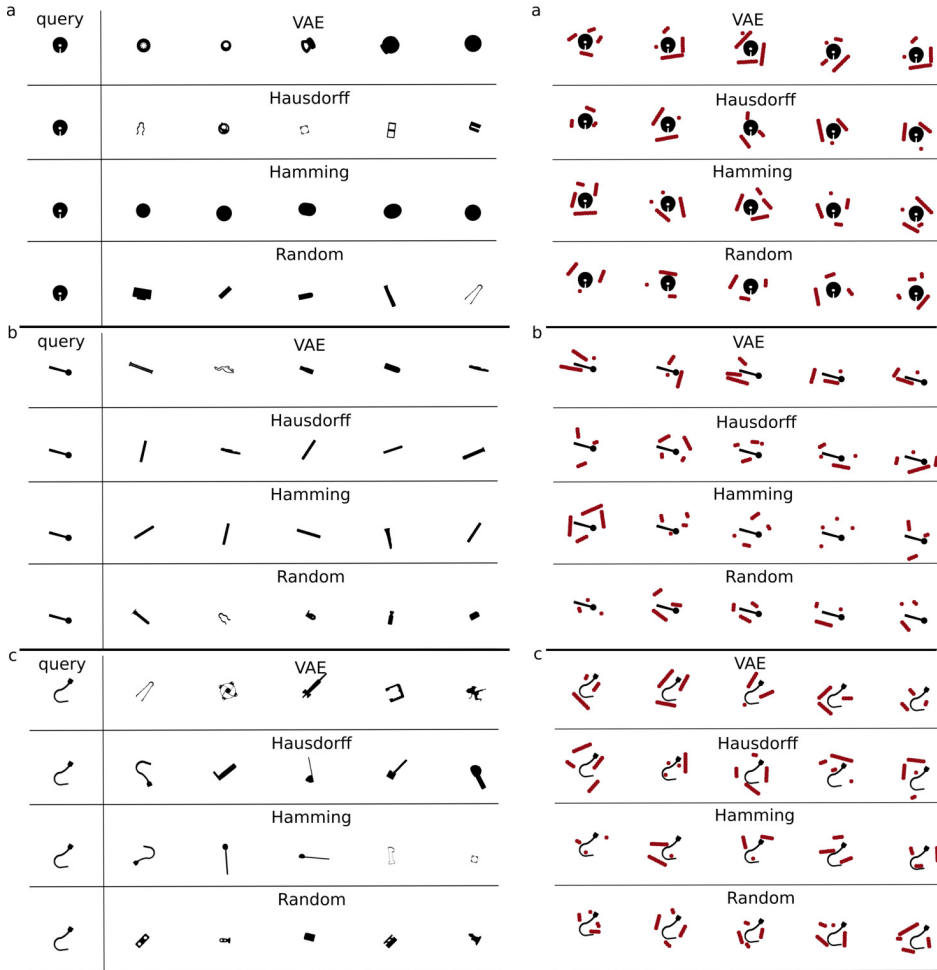


Figure A.18: On the left-hand side, we consider 3 different query objects washer(a), pin(b) and hook(c), and for each distance function visualize respective 5 closest objects from the training dataset; on the right-hand side, for each of the query object (a-c)) and each distance function, we visualize the acquired partial caging configurations.

For the VAE-model, it takes approximately 5 milliseconds to generate the latent representation, any subsequent distance query can then be performed in 0.005 milliseconds. The Hausdorff distance requires 0.5 milliseconds to compute, while the Hamming distance takes 1.7 milliseconds per distance calculation⁷.

⁷The time was measured on a Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.

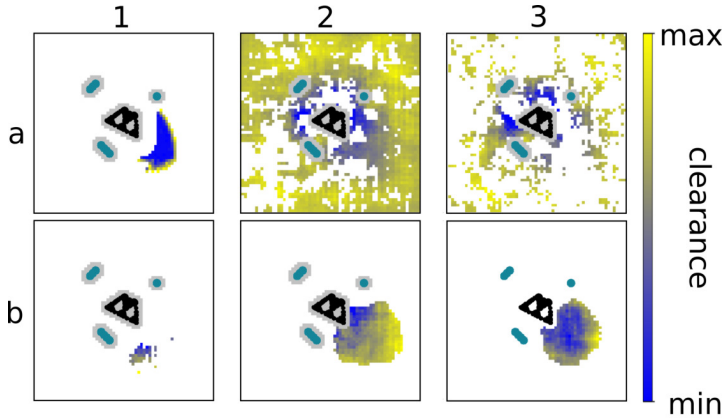


Figure A.19: Performance of *CageMaskNN* and *CageClearanceNN* given different numbers of training objects and evaluated on a single novel object. The top left (a1) displays the ground truth mask and clearance values for a fourth missing disc-shaped caging tool, a2: only 1 object is used for training, a3:10 objects are used for training, b1: 100 objects, b2: 1000 objects, b3: all 3048 objects are used for training. Note that the threshold had to be adjusted to 0.6 for the single object (a2) and 0.61 for the 10 object case (a3) to yield any discernible mask results at all.

Our experiments show that, while the VAE-induced similarity measure performs best in terms of finding collision-free caging tools placements, Hamming distance significantly outperforms it in terms of the quality of acquired partial cages. We did not observe a significant difference between Hausdorff distance and the VAE-induced distance. While Hamming distance appears to be better at capturing shape features that are relevant for cage acquisition task, it is the least efficient approach in terms of computation time. Furthermore, in our opinion, VAE-induced distance may be improved significantly if instead of using a general-purpose architecture we introduce task-specific geometric and topological priors.

9 Limitations and Challenges for Future Work

In this section, we discuss the main challenges of our work and the possible ways to overcome them.

9.1 Data generation challenges

One of the main challenges in this project is related to data generation: we need to densely sample the space of the caging tools' configurations, as well as the spaces of shapes of objects and caging tools. This challenge is especially significant when

using the *PC-general* dataset, as the space of possible caging tools configurations is large.

While the experimental evaluation indicates that the chosen network architecture is able to achieve low MSE on previously unseen objects, in applications one may want to train the network with either a larger distribution of objects, or a distribution of objects that are similar to the objects that will be encountered in practice.

In Fig.A.19, we illustrate how a lack of training data of sufficiently similar shapes can lead to poor performance of *CageMaskNN* and *CageClearanceNN*, for example, when only 1, 10, 100, or 1000 objects are used for training. Similarly, even when the networks are trained on the full training dataset of 3048 objects, the subtle geometric details of the partial caging region cannot be recovered for the novel test object, requiring more training data and further refinement of the approach.

9.2 Robustness under noise

In the cage acquisition scenario, the VAE-induced and Hamming distances work directly with images, and hence can be susceptible to noise. To evaluate this effect, we generate salt and pepper noise as well as Gaussian blur and analyse the performance of the VAE-induced and Hamming metrics under four different noise levels (0.005%, 0.01%, 0.05%, 0.1%) and four different kernel sizes (11x11, 21x21, 41x41, 61x61)⁸. Fig. A.20 shows the result of the top 3 retrieved objects for the hook object. Left column shows the query objects with respective disturbance. The next three columns depict the closest objects retrieved according to the VAE-induced metric, while the last three columns show the objects retrieved with Hamming metric.

Table A.3 reports the performance with respect to finding collision-free configurations, configurations belonging to \mathcal{C}_{cage} , and their average values of Q_{cl} . The results are averaged over 500 novel objects. We can see that the VAE-induced metric is affected by strong salt and pepper noise as the number of generated collision-free and partial caging configurations decreases. Furthermore, the resulting Q_{cl} of the generated partial cages increases, meaning it is easier to escape the cage. According to the experiment, the Hamming distance-based lookup is not significantly affected by salt and pepper noise. One explanation here may be that this kind of disturbance leads to a uniform increase of the Hamming distance for all objects. The Gaussian blur has a more negative effect on the Hamming distance lookup than the VAE-based lookup, as can be seen in the retrieved example objects in Fig. A.20. Table A.3 shows small decrease in the percentage of collision-free and partial caging configurations. Interestingly, the quality of the partial cages does not decrease.

⁸Note that sigma is calculated using the standard OpenCV [40] implementation ($\sigma = 0.3 \cdot ((ksize - 1) \cdot 0.5 - 1) + 0.8$).





















	query	VAE			Hamming		
a							
b							
							
							
							
c							
							
							
							

Figure A.20: Top three retrieval results for query images with different levels of disturbance for the VAE-induced and Hamming metric. *a* results without disturbance, *b* show retrieval for different level of salt and pepper noise, *c* retrieved objects when Gaussian blur is applied to query object (hook).

9.3 Real World Example and Future Work

As the VAE-framework just takes an image in order to propose suitable cages for a novel object, we showcase a concluding application example in Fig. A.21 where a novel object (a hand drill) is chosen as input to the VAE cage acquisition. The image is preprocessed by a simple threshold function to convert it to a black and white image, next the closest object from the dataset are found by comparing the distances in the latent space of the VAE and the three best partial caging configurations are retrieved and applied to the novel object.

Dist.	VAE			Hamming		
	\mathcal{C}_{free}	\mathcal{C}_{cage}	Q_{cl}	\mathcal{C}_{free}	\mathcal{C}_{cage}	Q_{cl}
no dist.	90.9 %	53.2 %	5.05	74.4 %	35.4 %	3.87
S&P 0.005	91.1 %	52.8 %	4.94	74.9 %	34.5 %	3.86
S&P 0.01	90.5 %	52.4 %	4.98	74.6 %	35.4 %	3.88
S&P 0.05	83.4 %	45.0 %	10.12	71.8 %	35.0 %	3.81
S&P 0.1	83.2 %	43.5 %	10.95	73.2 %	33.8 %	3.87
Gb 11x11	91.8 %	52.5 %	4.93	73.7 %	34.6 %	3.78
Gb 21x21	90.3 %	52.8 %	4.83	73.1 %	33.8 %	3.78
Gb 41x41	84.7 %	49.2 %	4.59	69.8 %	33.4 %	3.74
Gb 61x61	84.7 %	45.3 %	4.21	68.0 %	29.3 %	3.73

Table A.3: Performance for VAE-induced and Hamming metrics given different level of salt and pepper noise as well as Gaussian blur for different kernel sizes.



Figure A.21: Proposed partial cages using the VAE cage acquisition method. The novel object (hand drill) is feed into the cage acquisition and the best three cages from the closest object in the dataset are shown (in red).

In the future, we would like to extend our approach to 3-dimensional objects, As illustrated in Fig. A.22, partial cages may be a promising approach for transporting and manipulating 3D objects without the need for a firm grasp, and fast learning based approximations to analytic or planning based methods may be a promising direction for such partial 3D cages. Furthermore, we would also like to investigate the possibility of leveraging other caging verification methods such as [4] for our approach.



Figure A.22: An example for future partial caging in 3D. A complex object needs to be safely transported without the need to firmly grasp it.

10 Acknowledgements

This work has been supported by the Knut and Alice Wallenberg Foundation, Swedish Foundation for Strategic Research and Swedish Research Council.

References

1. A. Bicchi and V. Kumar, “Robotic grasping and contact: A review,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 348–353.
2. A. Rodriguez, M. T. Mason, and S. Ferry, “From caging to grasping,” *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012.
3. Z. McCarthy, T. Bretl, and S. Hutchinson, “Proving path non-existence using sampling and alpha shapes,” in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2563–2569.
4. A. Varava, J. F. Carvalho, F. T. Pokorny, and D. Kragic, “Free space of rigid objects: Caging, path non-existence, and narrow passage detection,” in *Workshop on Algorithmic Foundations of Robotics*, 2018.
5. T. Makapunyo, T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, “Measurement framework of partial cage quality,” in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2012, pp. 1812–1816.
6. J. Mahler, F. T. Pokorny, Z. McCarthy, A. F. van der Stappen, and K. Goldberg, “Energy-bounded caging: Formal definition and 2-d energy lower bound algorithm based on weighted alpha shapes,” *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 508–515, 2016.
7. J. Mahler, F. T. Pokorny, S. Niyaz, and K. Goldberg, “Synthesis of energy-bounded planar caging grasps using persistent homology,” *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 908–918, 2018.
8. A. Varava, M. C. Welle, J. Mahler, K. Goldberg, D. Kragic, and F. T. Pokorny, “Partial caging: A clearance-based definition and deep learning,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1533–1540.
9. E. Rimon and A. Blake, “Caging planar bodies by one-parameter two-fingered gripping systems,” *The International Journal of Robotics Research*, vol. 18(3), pp. 299–318, 1999.
10. P. Pipattanasomporn and A. Sudsang, “Two-finger caging of concave polygon,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 2137–2142.
11. M. Vahedi and A. F. van der Stappen, “Caging polygons with two and three fingers,” *The International Journal of Robotics Research*, vol. 27(11-12), pp. 1308–1324, 2008.

12. F. T. Pokorny, J. A. Stork, and D. Kragic, "Grasping objects with holes: A topological approach," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1100–1107.
13. J. A. Stork, F. T. Pokorny, and D. Kragic, "Integrated motion and clasp planning with virtual linking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.
14. J. A. Stork, F. T. Pokorny, and D. Kragic, "A topology-based object representation for clasping, latching and hooking," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 138–145.
15. A. Varava, D. Kragic, and F. T. Pokorny, "Caging grasps of rigid and partially deformable 3-d objects with double fork and neck features," *IEEE Transactions Robotics*, vol. 32, no. 6, pp. 1479–1497, 2016.
16. S. Makita and Y. Maeda, "3d multifingered caging: Basic formulation and planning," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2697–2702.
17. S. Makita, K. Okita, and Y. Maeda, "3d two-fingered caging for two types of objects: Sufficient conditions and planning," *International Journal of Mechatronics and Automation*, vol. 3, no. 4, pp. 263–277, 2013.
18. L. Zhang, Y. J. Kim, and D. Manocha, "Efficient cell labelling and path non-existence computation using c-obstacle query," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1246–1257, 2008.
19. W. Wan and R. Fukui, "Efficient planar caging test using space mapping," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 278–289, 2018.
20. J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2013.
21. D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv:1806.10293*, 2018.
22. J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Conference on Robot Learning*, 2017, pp. 515–524.
23. A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *arXiv:1710.01330*, 2017.
24. D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4304–4311.
25. I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

26. A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.
27. S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.
28. L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.
29. K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4243–4250.
30. M. Gualtieri, A. Ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 598–605.
31. E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4461–4468.
32. J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.
33. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
34. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.
35. D. Kinga and J. Adam, "A method for stochastic optimization int," in *Conf. on Learning Representations (ICLR.)*, 2015.
36. S. Lembecke, "Chipmunk 2d physics engine," *Howling Moon Software*, 2013.
37. B. Dai and D. Wipf, "Diagnosing and enhancing vae models," *arXiv preprint arXiv:1903.05789*, 2019.
38. A. A. Taha and A. Hanbury, "An Efficient Algorithm for Calculating the Exact Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2153–2163, 2015.
39. R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.
40. G. Bradski, "The OpenCV library," vol. 25, no. 11, pp. 120, 122–125, Nov. 2000. [Online]. Available: http://www.ddj.com/ftp/2000/2000_11/opencv.txt

Paper B

Paper B

Enabling Visual Action Planning for Object Manipulation through Latent Space Roadmap

Martina Lippi*, Petra Poklukar*, Michael C. Welle*, Anastasia Varava, Hang Yin, Alessandro Marino, and Danica Kragic

Abstract

We present a framework for visual action planning of complex manipulation tasks with high-dimensional state spaces, focusing on manipulation of deformable objects. We propose a Latent Space Roadmap (LSR) for task planning which is a graph-based structure globally capturing the system dynamics in a low-dimensional latent space. Our framework consists of three parts: (1) a Mapping Module (MM) that maps observations given in the form of images into a structured latent space extracting the respective states as well as generates observations from the latent states, (2) the LSR which builds and connects clusters containing similar states in order to find the latent plans between start and goal states extracted by MM, and (3) the Action Proposal Module that complements the latent plan found by the LSR with the corresponding actions. We present a thorough investigation of our framework on simulated box stacking and rope/box manipulation tasks, and a folding task executed on a real robot.

1 Introduction

In task and motion planning, it is common to assume that the geometry of the scene is given as input to the planner. In contrast, modern representation learning methods are able to automatically extract state representations directly from

* Authors contributed equally, listed in alphabetical order.

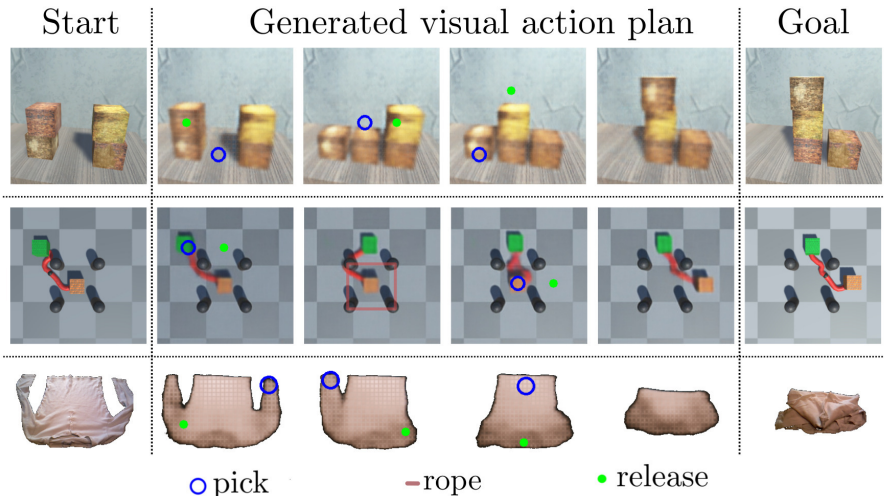


Figure B.1: Examples of visual action plans for a stacking task (top), a rope/box manipulation task (middle) and a shirt folding task (bottom).

high-dimensional raw observations, such as images or video sequences [1]. This is especially useful in complex scenarios where explicit analytical modeling of states is challenging, such as in manipulation of *highly deformable* objects which is recently gaining increasing attention by the research community [2, 3].

Unsupervised State Representation Learning. Given raw observations, state representation learning is commonly performed in an unsupervised way using for example Autoencoders (AEs) [4] or Variational Autoencoders (VAEs) [5]. In these frameworks, two neural networks – an encoder and a decoder – are jointly trained to embed the input observation into a low-dimensional latent space, and to reconstruct it given a latent sample. The resulting latent space can be used as a low-dimensional representation of the state space, where the encoder acts as a map from a high-dimensional observation (an image) into the lower-dimensional state (a latent vector).

However, to be useful for planning, it is desirable to have a particular structure in the latent space: states that are similar should be encoded close to each other, while different states should be separated. Such information does not always coincide with the similarity of the respective images: two observations can be significantly different with respect to a pixel-wise metric due to task-irrelevant factors of variation such as changes in the lighting conditions and texture, while the underlying state of the system (e.g., the shape and the pose of the objects) may be identical. The opposite is also possible: two observations may be relatively close in the image space, because the respective change in the configuration of the scene

does not significantly affect the pixel-wise metric, while from the task planning perspective the two states are fundamentally different.

Challenges of State Representation Learning for Planning. For planning, the system dynamics should also be captured in the latent space. We therefore identify three main challenges when modeling the state space representation for planning: *i*) it needs to be low dimensional, while containing the relevant information from high-dimensional observations; *ii*) it needs to properly reflect similarities between states; and *iii*) it needs to efficiently capture feasible transitions between states allowing to perform complex tasks such as deformable object manipulation.

In this work, we address *i*) by extracting the low-dimensional states directly from images of the scene through a Mapping Module (MM). For this, we deploy a VAE framework and compare it to AE. We address *ii*) by explicitly encouraging the encoder network to map the observations that correspond to different states further away from each other despite their visual similarity. This is done by providing a weak supervision signal: we record a small number of actions between observation pairs, and mark the observations as “same” or “different” depending on whether or not an action is needed to bring the system from one state to the successor one. We use this action information in an additional loss term to structure the latent space accordingly. Finally, we tackle *iii*) by building the Latent Space Roadmap (LSR), which is a graph-based structure in the latent space used to plan a manipulation sequence given a start and goal image of the scene. The nodes of this graph are associated with the system states, and the edges model the actions connecting them. For example, as shown in Fig. B.1, these actions can correspond to moving a box or a rope, or folding a shirt. We identify the regions containing the same underlying states using hierarchical clustering [6] which accounts for differences in shapes and densities of these regions. The extracted clusters are then connected using the weak supervision signals. Finally, the action specifics are obtained from the Action Proposal Module (APM). In this way, we capture the global dynamics of the state space in a data-efficient manner without explicit state labels, which allows us to learn a state space representation for complex *long-horizon* tasks.

Contributions. Our contributions can be summarized as follows: *i*) we define the Latent Space Roadmap that enables to generate visual action plans based on weak supervision; *ii*) we introduce an augmented loss function with dynamic parameter to favourably structure the latent space; *iii*) we validate our framework on simulated box stacking tasks involving rigid objects, a combined rope and box manipulation task involving both deformable and rigid objects, and on a real-world T-shirt folding task involving deformable objects. Complete details can be found on the website¹.

This work is an extensively revised version of our earlier conference paper [7], where we first introduced the notion of Latent Space Roadmap. The main novelties of the present work with respect to [7] are:

¹<https://visual-action-planning.github.io/lsr-v2/>

1. Extension of the LSR building algorithm with an outer optimisation loop improving its performance;
2. New training approach for the MM with a dynamic adjustment of the key hyperparameter used in the additional loss term;
3. Large scale simulation campaigns investigating the effect of the additional loss term and hyperparameter choices;
4. Restructuring of the framework into three main components leading to a more modular setup;
5. Introduction of a more challenging box stacking task and a task involving manipulation of a rope and two boxes, enabling a thorough ablation study on all components of our framework;
6. Comparison with the state-of-the-art solutions in [8] and [9] on the simulation tasks as well as comparison of the improved framework with its predecessor [7] on the T-shirt folding task performed on a real robot;
7. Comparison with other potentially suitable clustering algorithms used to build the LSR;
8. Comparison of VAE and AE for the mapping module;
9. Comparison of different realizations of the APM.

2 Related Work

The state representation, in terms of shape and poses of objects in the robot’s workspace, is generally assumed to be known in task and motion planning and represents an input to the planner. As an example, robot and grasping knowledge was exploited in [10] to carry out sequential manipulation tasks, while semantic maps were designed in [11] to integrate spatial and semantic information and perform task planning. In addition, the notion of belief space, representing all possible distributions over the robot state space, was introduced in [12] to tackle partially observable control problems. Alternatively, sampling-based planning approaches, such as Rapidly exploring Random Tree (RRT) or Rapidly exploring Dense Tree (RDT) [13], can be adopted to probe the state space when an explicit representation of it is difficult to achieve.

However, traditional planning approaches, like the ones mentioned above, are generally prone to fail when high-dimensional states, such as images, are involved in the system [14]. For this reason, data-driven low-dimensional latent space representations for planning are gaining increasing attention, which, nonetheless, introduce problems to properly capture the global structure and dynamics of the system, as discussed in Sec. 1. To tackle these, imitation learning can be leveraged as

in [15, 16]. In particular, a latent space Universal Planning Network was designed in [15] to embed differentiable planning policies. The process is learned in an end-to-end fashion from imitation learning and gradient descent is used to find optimal trajectories. A controller based on random forests was built in [16] to perform cloth manipulation assuming the presence of optimal demonstrations provided by experts. Alternatively, a motion planning network with active learning procedure was developed in [17] to reduce the data for training and actively ask for expert demonstrations only when needed. In particular, the approach relies on an encoder network, that maps the robot surrounding environment in a low-dimensional state space, and on a prediction network, that is iteratively queried to provide states in the planned path on the basis of environment encoding and the robot initial and goal states.

In contrast to imitation learning methods, self-supervised learning approaches can be applied to mitigate the reduced generalization capabilities of the former at the expense of a costly data collection phase. Existing methods leverage physical simulators to acquire interaction data and then adapt skills from simulation to reality [18]. New simulation techniques have also opened up research on manipulating cloth-like objects involving multiple actions [19, 20]. However, these methods are often limited to handling objects of a simple shape, such as a square cloth, and simplify the interaction modeling with kinematic constraints [18, 19, 20]. Simulating complex dynamics of deformable objects/robot interaction remains an open problem. This was the motivation in [21] which learned folding directly on the real robotic platforms by meticulously designing convolutional observation features and a discrete action space, managing to generalize to objects of untrained sizes. Still, the scalability of collecting real interactive data undermines the applicability of self-supervised learning approaches. This problem becomes even more severe when dealing with long-horizon task planning, where the range of possible variations in the state space is large. In this context, a framework for global search in latent space was designed in [22] which is based on three components: *i*) a latent state representation, *ii*) a network to approximate the latent space dynamics, and *iii*) a collision checking network. Motion planning is then performed directly in the latent space by an RRT-based algorithm. Embed-to-Control (E2C) [23] pioneers in learning a latent linear dynamical model for planning continuous actions. Variational inference is used to infer a latent representation and dynamical system parameters that can reconstruct a sequence of images. The extracted latent variables were shown to be well aligned to the underlying states in a visual inverted pendulum domain. In addition to estimating transition and observation models, [9] proposed a deep planning network which also learns a reward function in the latent space. Then, the latter is used to find viable trajectories resorting to a Model Predictive Control (MPC) approach. A comparison between our method and a baseline inspired by this approach can be found in Sec. 9.3.1. Reinforcement Learning (RL) using self play was employed in [24], where a VAE encodes trajectories into the latent space that is optimized to minimize the KL-divergence between proposed latent plans and those that have been encountered during self-play exploration. In contrast to

using full plans, we optimized the structure of the latent space using pairs of states. The manipulation of a deformable rope from an initial start state to a desired goal state was analyzed in [25]. It builds upon [26] and [27] where 500 hours worth of data collection were used to learn the rope inverse dynamics, and then produced an *understandable* visual foresight plan containing the intermediate steps to deform the rope using a Context Conditional Causal InfoGAN (C^3 IGAN). Similarly, contrastive learning was used in [28] to learn a predictive model in the latent space, which is used to plan rope and cloth flattening actions. Goal-conditioning was then introduced in [29] to carry out long-horizon visual planning by reducing the search space and performing hierarchical optimization. Visual foresight was also realized in [30] where a video prediction model based on Long-Short Term Memory blocks was employed to predict stochastic pixel flow from frame to frame. Trained on video, action and state sequences, the model provides an RGB prediction of the scene that is then used to perform visual model predictive control. The data was collected using ten identical real world setups with different camera angles. In this work, instead of performing imitation learning from experts or exploring the latent space in a self-supervised manner, we leverage a weak supervision given by demonstrated actions in the dataset to capture the global structure of the state space and its dynamics in a data-efficient manner.

Finally, the idea of employing graphs as data representation has recently been investigated in several works. In [31], the authors introduced contrastive learning for structured world models. An end-to-end framework was presented that uses a Convolutional Neural Network (CNN) extractor to segment objects from the given scene observations and a Multi-Layer Perceptron encoder to produce their representations. A graph neural network (GNN) was then learned to model the relations and transitions given the representations of objects in the scene. Moreover, as graphs naturally decompose a long-horizon task into short-horizon tasks, authors of [32] improved RL performance on long-horizon tasks by building a graph whose nodes are the states contained in the replay buffer. Combining RL with the idea of connecting states in the latent space via a graph was proposed in Semi-Parametric Topological Memory (SPTM) framework [8], where an agent explores the environment and encodes observations into a latent space using a retrieval network. Each encoded observation forms a unique node in a memory graph built in the latent space. This graph is then used to plan an action sequence from a start to a goal observation using a locomotion network. As discussed in Sec. 9.3.1, where we compare our method with the SPTM framework, the latter is optimized for the continuous domain, having action/observation trajectories as input and building on the assumption that each observation is mapped to a unique latent code. Finally, the graph representation was also used in [33] for learning block stacking. The scene/object relations are expected to be captured by a GNN so as to boost the efficiency of curriculum learning and to generalize to stacking untrained number of blocks. In this work, we use graphs to capture the structure and the dynamics of the system, where nodes represent states of the system and edges the possible transitions between them.

3 Problem Statement and Notation

Variable	Meaning
\mathcal{I}	Space of observations, i. e. , images
\mathcal{U}	Space of actions
\mathcal{Z}	Low-dimensional latent space
P_I, P_u, P_z	Planned sequence of images, actions and latent states from assigned start and goal observations, respectively
\mathcal{Z}_{sys}^i	Covered region i of the latent space
\mathcal{Z}_{sys}	Overall covered region of the latent space
ρ	Specifics of the action that took place between two images I_1 and I_2
$\mathcal{T}_I, \mathcal{T}_z$	Datasets containing image tuples (I_1, I_2, ρ) and their embeddings (z_1, z_2, ρ) , respectively
ξ	Latent mapping function from \mathcal{I} to \mathcal{Z}
ω	Observation generator function from \mathcal{Z} to \mathcal{I}
d_m	Minimum distance encouraged among action pairs in the latent space
p	Metric L_p
τ	Clustering threshold for LSR building
c_{max}	Maximum number of connected components of the LSR
$N_{\varepsilon_z}(z)$	The ε_z -neighbourhood of a covered state z containing same covered states
ε^i	ε_z associated with all the states in the covered region \mathcal{Z}_{sys}^i , i. e. $\varepsilon^i = \varepsilon_z \forall z \in \mathcal{Z}_{sys}^i$

Table B.1: Main notations introduced in the paper.

The goal of visual action planning, also referred to as “*visual planning and acting*” in [25], can be formulated as follows: given start and goal images, generate a path as a sequence of images representing intermediate states and compute dynamically valid actions between them. We now formalize the problem and provide notation in Table B.1.

Let \mathcal{I} be the space of all possible observations of the system’s states represented as images with fixed resolution and let \mathcal{U} be the set of possible control inputs or actions.

Definition 5. A visual action plan consists of a visual plan represented as a sequence of images $P_I = \{I_{start} = I_0, I_1, \dots, I_N = I_{goal}\}$ where $I_{start}, I_{goal} \in \mathcal{I}$ are images capturing the underlying start and goal states of the system, respectively, and an action plan represented as a sequence of actions $P_u = \{u_0, u_1, \dots, u_{N-1}\}$ where $u_n \in \mathcal{U}$ generates a transition between consecutive states contained in the observations I_n and I_{n+1} for each $n \in \{0, \dots, N-1\}$.

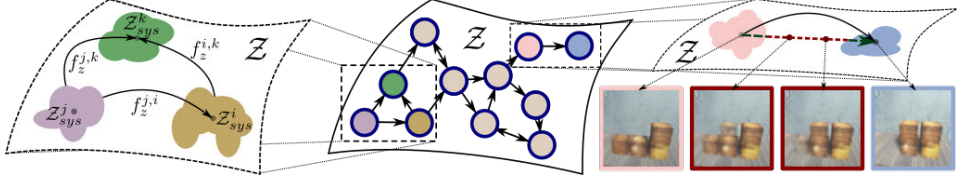


Figure B.2: Illustrative representation of the latent space \mathcal{Z} . In the middle, possible transitions (arrows) between covered regions (sketched with circles) are shown. On the left, details of the covered regions with different shapes and representative points are provided. On the right, observations from a box stacking tasks are shown. In detail, the ones obtained from covered regions (in pink and blue) contain meaningful task states, while the ones generated from not covered regions (in red) show fading boxes that do not represent possible states of the system.

To retrieve the underlying states represented in the observations as well as to reduce the complexity of the problem we map \mathcal{I} into a lower-dimensional latent space \mathcal{Z} such that each observation $I_n \in \mathcal{I}$ is encoded as a point $z_n \in \mathcal{Z}$ extracting the state of the system captured in the image I_n . We call this map a *latent mapping* and denote it by $\xi : \mathcal{I} \rightarrow \mathcal{Z}$. In order to generate visual plans, we additionally assume the existence of a mapping $\omega : \mathcal{Z} \rightarrow \mathcal{I}$ called *observation generator*.

Let $\mathcal{T}_I = \{I_1, \dots, I_M\} \subset \mathcal{I}$ be a finite set of input observations inducing a set of *covered states* $\mathcal{T}_z = \{z_1, \dots, z_M\} \subset \mathcal{Z}$, i. e., $\mathcal{T}_z = \xi(\mathcal{T}_I)$. In order to identify a set of unique covered states, we make the following assumption on \mathcal{T}_z .

Assumption 2. *Let $z \in \mathcal{T}_z$ be a covered state. Then there exists $\varepsilon_z > 0$ such that any other state z' in the ε_z -neighborhood $N_{\varepsilon_z}(z)$ of z can be considered as the same underlying state.*

This allows both generating a valid visual action plan and taking into account the uncertainty induced by imprecisions in action execution. Let

$$\mathcal{Z}_{sys} = \bigcup_{z \in \mathcal{T}_z} N_{\varepsilon_z}(z) \subset \mathcal{Z} \quad (\text{B.1})$$

be the union of ε_z -neighbourhoods of the covered states $z \in \mathcal{T}_z$. Given \mathcal{Z}_{sys} , a visual plan can be computed in the latent space using a *latent plan* $P_z = \{z_{start} = z_0, z_1, \dots, z_N = z_{goal}\}$, where $z_n \in \mathcal{Z}_{sys}$, which is then decoded with the observation generator ω into a sequence of images.

To obtain a valid visual plan, we study the structure of the space \mathcal{Z}_{sys} which in general is not path-connected. As we show in Fig. B.2 on the right, linear interpolation between two states z_1 and z_2 in \mathcal{Z}_{sys} may result in a path containing points from $\mathcal{Z} - \mathcal{Z}_{sys}$ that do not correspond to covered states of the system and are therefore not guaranteed to be meaningful. To formalize this, we define an

equivalence relation in \mathcal{Z}_{sys}

$$z \sim z' \iff z \text{ and } z' \text{ are path-connected in } \mathcal{Z}_{sys}, \quad (\text{B.2})$$

which induces a partition of the space \mathcal{Z}_{sys} into m equivalence classes $[z_1], \dots, [z_m]$. Each equivalence class $[z_i]$ represents a path-connected component of \mathcal{Z}_{sys}

$$\mathcal{Z}_{sys}^i = \bigcup_{z \in [z_i]} N_{\varepsilon_z}(z) \subset \mathcal{Z}_{sys} \quad (\text{B.3})$$

called *covered region*. To connect the covered regions, we define a set of transitions between them:

Definition 6. A transition function $f_z^{i,j} : \mathcal{Z}_{sys}^i \times \mathcal{U} \rightarrow \mathcal{Z}_{sys}^j$ maps any point $z \in \mathcal{Z}_{sys}^i$ to an equivalence class representative $z_{sys}^j \in \mathcal{Z}_{sys}^j$, where $i, j \in \{1, 2, \dots, m\}$ and $i \neq j$.

Equivalence relation (B.2) and Assumption 2 imply that two distinct observations I_1 and I_2 which are mapped into the same covered region \mathcal{Z}_{sys}^i contain the same underlying state of the system, and can be represented by the same equivalence class representative z_{sys}^i . Given a set of covered regions \mathcal{Z}_{sys}^i in \mathcal{Z}_{sys} and a set of transition functions connecting them we can approximate the global transitions of \mathcal{Z}_{sys} as shown in Fig. B.2 on the left. To this end, we define a Latent Space Roadmap (see Fig. B.2 in the middle):

Definition 7. A Latent Space Roadmap is a directed graph $\text{LSR} = (\mathcal{V}_{\text{LSR}}, \mathcal{E}_{\text{LSR}})$ where each vertex $v_i \in \mathcal{V}_{\text{LSR}} \subset \mathcal{Z}_{sys}$ for $i \in \{1, 2, \dots, m\}$ is an equivalence class representative of the covered region $\mathcal{Z}_{sys}^i \subset \mathcal{Z}_{sys}$, and an edge $e_{i,j} = (v_i, v_j) \in \mathcal{E}_{\text{LSR}}$ represents a transition function $f_z^{i,j}$ between the corresponding covered regions \mathcal{Z}_{sys}^i and \mathcal{Z}_{sys}^j for $i \neq j$. Moreover, weakly connected components of an LSR are called graph-connected components.

4 Methodology

We first present the structure of the training dataset and then provide an overview of the approach.

4.1 Training Dataset

We consider a training dataset \mathcal{T}_I consisting of generic tuples of the form (I_1, I_2, ρ) where $I_1 \subset \mathcal{I}$ is an image of the start state, $I_2 \subset \mathcal{I}$ an image of the successor state, and ρ a variable representing the action that took place between the two observations. Here, an action is considered to be a *single* transformation that produces any consecutive state represented in I_2 different from the start state in I_1 , i. e. , ρ cannot be a composition of several transformations. On the contrary, we

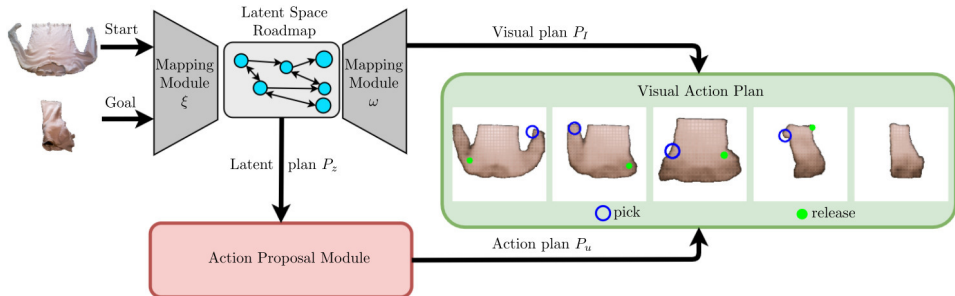


Figure B.3: Overview of the proposed method. Start and goal images (on the left) are mapped to the latent space \mathcal{Z} by the latent mapping ξ of the Mapping Module. A latent plan is then found with the Latent Space Roadmap (cyan circles and arrows) and is *decoded* to a visual plan using the observation generator ω of the Mapping Module. The Action Proposal Module (red) proposes suitable actions to achieve the transitions between states in the visual plan. The final result is a *visual action plan* (green) from start to goal. A re-planning step can also be added after every action to account for execution uncertainties as shown in Fig. B.12 on the T-shirt folding task.

say that no action was performed if images I_1 and I_2 are observations of the same state, i. e. , if $\xi(I_1) \sim \xi(I_2)$ with respect to the equivalence relation (B.2). The variable $\rho = (a, u)$ consists of a binary variable $a \in \{0, 1\}$ indicating whether or not an action occurred as well as a variable u containing the task-dependent action-specific information. The latter, if available, is used to infer the transition functions $f_z^{i,j}$. We call a tuple $(I_1, I_2, \rho = (1, u))$ an *action pair* and $(I_1, I_2, \rho = (0, u))$ a *no-action pair*. For instance, Fig. B.4 shows an example of an action pair (top row) and a no-action pair (bottom row) for the folding task. In this case, the action specifics u contain the pick and place coordinates to achieve the transition from the state captured by I_1 to the state captured by I_2 , while the no-action pair images are *different* observations of the same underlying state of the system represented by slight perturbations of the sleeves. When the specifics of an action u are not needed, we omit them from the tuple notation and simply write (I_1, I_2, a) . By abuse of notation, we sometimes refer to an observation I contained in any of the training tuples as $I \in \mathcal{T}_I$. Finally, we denote by \mathcal{T}_z the encoded training dataset \mathcal{T}_I consisting of latent tuples (z_1, z_2, ρ) obtained from the input tuples $(I_1, I_2, \rho) \in \mathcal{T}_I$ by encoding the inputs I_1 and I_2 into the latent space \mathcal{Z}_{sys} with the latent mapping ξ . The obtained states $z_1, z_2 \in \mathcal{Z}_{sys}$ are called *covered states*.

Remark 1. *The dataset \mathcal{T}_I is not required to contain all possible action pairs of the system but only a subset of them that sufficiently cover the dynamics, which makes our approach data efficient.*

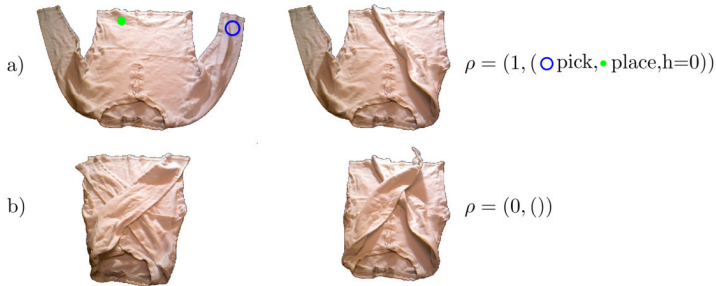


Figure B.4: Example of action (a) and no-action (b) pairs in folding task.

4.2 System Overview

Generation of visual action plans consists of three components visualized in Fig. B.3:

- **Mapping Module (MM)** used to both extract a low-dimensional representation of a state represented by a given observation, and to generate an exemplary observation from a given latent state (Sec. 5);
- **Latent Space Roadmap (LSR)** built in the low dimensional latent space and used to plan (Sec. 6);
- **Action Proposal Module (APM)** used to predict action specifics for executing a latent plan found by the LSR (Sec. 7).

The MM consists of the latent mapping $\xi : \mathcal{I} \rightarrow \mathcal{Z}$ and the observation generator $\omega : \mathcal{Z} \rightarrow \mathcal{I}$. To find a visual plan between a given start observation I_{start} and goal observation I_{goal} , the latent mapping ξ first extracts the corresponding lower-dimensional representations z_{start} and z_{goal} of the underlying start and goal states, respectively. Ideally, ξ should perfectly extract the underlying state of the system such that different observations containing the same state are mapped into the same latent point. In practice, however, the unknown true latent embedding ξ is *approximated* with a neural network which implies that different observations containing the same state could be mapped to different latent points. In order to perform planning in \mathcal{Z} , we thus build the LSR which is a graph-based structure identifying the latent points belonging to the same underlying state and approximating the system dynamics. This enables finding the latent plans P_z between the extracted states z_{start} and z_{goal} . For the sake of interpretability, latent plans P_z are *decoded* into visual plans P_I , consisting of a sequence of images, by the observation generator ω .

We complement the generated visual plan P_I with the action plan P_u produced by the APM, which proposes an action u_i that achieves the desired transition $f_z^{i,i+1}(z_i, u_i) = z_{i+1}$ between each pair (z_i, z_{i+1}) of consecutive states in the latent plan P_z found by the LSR.

The visual action plan produced by the three components can be executed by any suitable framework.

Remark 2. *If open loop execution is not sufficient for the task, as for deformable object manipulation, a re-planning step can be added after every action. This implies that a new visual action plan is produced after the execution of each action until the goal is reached. A visualization of the re-planning procedure is shown in Fig. B.12 on the T-shirt folding task presented in Sec. 10.*

Remark 3. *Our method is able to generate a sequence of actions $\{u_0, \dots, u_{N-1}\}$ to reach a goal state in I_N from a given start state represented by I_0 , even though the tuples in the input dataset \mathcal{T}_I only contain single actions u that represent the weak supervision signals.*

5 Mapping Module (MM)

The mappings $\xi : \mathcal{I} \rightarrow \mathcal{Z}$ and $\omega : \mathcal{Z} \rightarrow \mathcal{I}$ as well as the low-dimensional space \mathcal{Z} can be realized using any encoder-decoder based algorithms, for example VAEs, AEs or Generative Adversarial Networks (GANs) combined with an encoder network. The primary goal of MM is to find the best possible approximation ξ such that the structure of the extracted states in the latent space \mathcal{Z} resembles the one corresponding to the unknown underlying system. The secondary goal of MM is to learn an observation generator ω which enables visual interpretability of the latent plans. Since the quality of these depends on the structure of the latent space \mathcal{Z} , we leverage the action information contained in the binary variable a of the training tuples (I_1, I_2, a) to improve the quality of the latent space. We achieve this by introducing a contrastive loss term [34] which can be easily added to the loss function of any algorithm used to model the MM.

More precisely, we introduce a general *action* term

$$\mathcal{L}_{action}(I_1, I_2) = \begin{cases} \max(0, d_m - \|z_1 - z_2\|_p) & \text{if } a = 1 \\ \|z_1 - z_2\|_p & \text{if } a = 0 \end{cases} \quad (\text{B.4})$$

where $z_1, z_2 \in \mathcal{Z}_{sys}$ are the latent encodings of the input observations $I_1, I_2 \in \mathcal{T}_I$, respectively, d_m is a hyperparameter, and the subscript $p \in \{1, 2, \infty\}$ denotes the metric L_p . The action term \mathcal{L}_{action} naturally imposes the formulation of the covered regions \mathcal{Z}_{sys}^i in the latent space. On one hand, it encodes identical states contained in the no-action pairs close by. On the other hand, it encourages different states to be encoded in separate parts of the latent space via the hyperparameter d_m .

As we experimentally show in Sec. 9.2.1, the choice of d_m has a substantial impact on the latent space structure. Therefore, we propose to learn its value *dynamically* during the training of the MM. In particular, d_m is increased until the separation of action and no-action pairs is achieved. Starting from 0 at the beginning of the training, we increase d_m by Δd_m every k th epoch as long as the

maximum distance between no-action pairs is larger than the minimum distance between action pairs. The effect of dynamically increasing d_m is shown in Fig. B.5 where we visualize the distance $\|z_1 - z_2\|_1$ between the latent encodings of every action training pair (in blue) and no-action training pair (in green) obtained at various epochs during training on a box stacking task. It can be clearly seen that the parameter d_m is increased as long as there is an intersection between action and no-action pairs. Detailed investigation of this approach as well as its positive effects on the structure of the latent space are provided in Sec. 9.2.1. Note that the dynamic adaptation of the parameter d_m eliminates the need to predetermine its value as in our previous work [7].

We use a VAE such that its latent space represents the space \mathcal{Z} , while the encoder and decoder networks realize the mappings ξ and ω , respectively. We validate this choice in Sec. 9.2.3 by comparing it to AE. In the following, we first provide a brief summary of the VAE framework [5, 35] and then show how the action term can be integrated into its training objective. Let $I \subset \mathcal{T}_I$ be an input image, and let z denote the unobserved latent variable with prior distribution $p(z)$. The VAE model consists of encoder and decoder neural networks that are jointly optimized to represent the parameters of the approximate posterior distribution $q(z|I)$ and the likelihood function $p(I|z)$, respectively. In particular, VAE is trained to minimize

$$\mathcal{L}_{vae}(I) = E_{z \sim q(z|I)}[\log p(I|z)] + \beta \cdot D_{KL}(q(z|I) || p(z)) \quad (\text{B.5})$$

with respect to the parameters of the encoder and decoder neural networks. The first term influences the quality of the reconstructed samples, while the second term, called Kullback-Leibler (KL) divergence term, regulates the structure of the latent space. The trade-off between better reconstructions or a more structured latent space is controlled by the parameter β , where using a $\beta > 1$ favors the latter [36, 37]. The action term (B.4) can be easily added to the VAE loss (B.5) as follows:

$$\mathcal{L}(I_1, I_2) = \frac{1}{2}(\mathcal{L}_{vae}(I_1) + \mathcal{L}_{vae}(I_2)) + \gamma \cdot \mathcal{L}_{action}(I_1, I_2) \quad (\text{B.6})$$

where $I_1, I_2 \subset \mathcal{T}_I$ and the parameter γ controls the influence of the distances among the latent encodings on the latent space structure. Note that the same procedure applies for integrating the action term (B.4) into any other framework that models the MM.

6 Latent Space Roadmap (LSR)

The Latent Space Roadmap, defined in *Definition 7*, is built in the latent space \mathcal{Z} obtained from the MM. LSR is a graph that enables planning in the latent space which identifies sets of latent points associated with the same underlying state and viable transitions between them. Each node in the roadmap is associated with a covered region \mathcal{Z}_{sys}^i . Two nodes are connected by an edge if there exists an action pair $(I_1, I_2, \rho = (1, u_1))$ in the training dataset \mathcal{T}_I such that the transition $f_z^{1,2}(z_1, u_1) = z_2$ is achieved in \mathcal{Z}_{sys} .

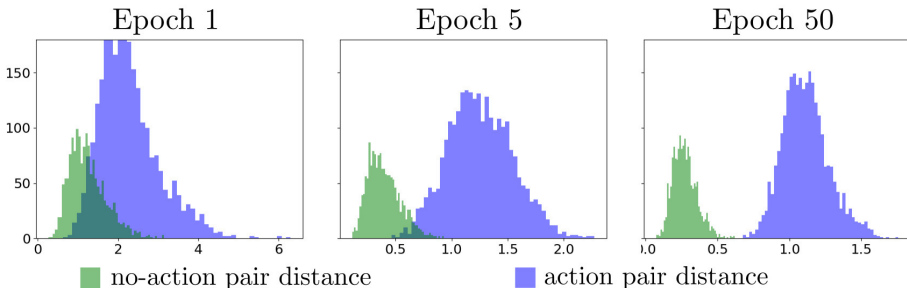


Figure B.5: An example showing histograms of distances $\|z_1 - z_2\|_1$ for latent action (in blue) and no-action pairs (in green) obtained at epochs 1, 5 and 50 during the training of VAE on the hard box stacking task (more details in Sec. 9). The figure shows the separation of the action and no-action distances induced by dynamically increasing the minimum distance d_m in \mathcal{L}_{action} .

The LSR building procedure is summarized in Algorithm 2 and discussed in Sec. 6.1. It relies on a clustering algorithm that builds the LSR using the encoded training data \mathcal{T}_z and a specified metric L_p as inputs. The input parameter τ is inherited from the clustering algorithm and we automatically determine it using the procedure described in Sec. 6.2.

6.1 LSR Building

Algorithm 2 consists of three phases. In Phase 1 (lines 1.1 – 1.5), we build a *reference* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ induced by \mathcal{T}_z and visualized on the top left of Fig. B.6. Its set of vertices \mathcal{V} is the set of all the latent states in \mathcal{T}_z , while edges exist only among the latent action pairs. It serves as a look-up graph to preserve the edges that later induce the transition functions $f_z^{i,j}$.

In Phase 2, Algorithm 2 identifies the covered regions $\mathcal{Z}_{sys}^i \subset \mathcal{Z}_{sys}$. We achieve this by first clustering the training samples and then retrieving the covered regions from these clusters. We start by performing agglomerative clustering [6] on the encoded dataset \mathcal{T}_z (line 2.1). Agglomerative clustering is a hierarchical clustering scheme that starts from single nodes of the dataset and merges the closest nodes, according to a dissimilarity measure, step by step until only one node remains. It results in a *stepwise dendrogram* M , depicted in the top right part of Fig. B.6, which is a tree structure visualizing the arrangement of data points in clusters with respect to the level of dissimilarity between them. We choose to measure this inter-cluster dissimilarity using the *unweighted average* distance between points in each cluster, a method also referred to as UPGMA [38]. More details about other possible clustering algorithms and dissimilarity measures are discussed in Sec. 9.3.4. Next, the dissimilarity value τ , referred to as *clustering threshold*, induces the set

Algorithm 2 LSR building

Require: Dataset \mathcal{T}_z , metric L_p , clustering threshold τ

Phase 1

- 1: init graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) := (\{\}, \{\})$
- 2: **for each** $(z_1, z_2, a) \in \mathcal{T}_z$ **do**
- 3: $\mathcal{V} \leftarrow$ create nodes z_1, z_2
- 4: **if** $a = 1$ **then**
- 5: $\mathcal{E} \leftarrow$ create edge (z_1, z_2)

Phase 2

- 1: $M \leftarrow$ Average-Agglomerative-Clustering(\mathcal{T}_z, L_p) [6]
- 2: $\mathcal{W} \leftarrow$ get-Disjoint-Clusters(M, τ)
- 3: $\mathcal{Z}_{sys} \leftarrow \{\}$
- 4: **for each** $\mathcal{W}^i \in \mathcal{W}$ **do**
- 5: $\varepsilon^i \leftarrow$ get-Cluster-Epsilon(\mathcal{W}^i)
- 6: $\mathcal{Z}_{sys}^i := \cup_{w \in \mathcal{W}^i} N_{\varepsilon^i}(w)$
- 7: $\mathcal{Z}_{sys} := \mathcal{Z}_{sys} \cup \{\mathcal{Z}_{sys}^i\}$

Phase 3

- 1: init graph LSR = $(\mathcal{V}_{LSR}, \mathcal{E}_{LSR}) := (\{\}, \{\})$
- 2: **for each** $\mathcal{Z}_{sys}^i \in \mathcal{Z}_{sys}$ **do**
- 3: $w^i := \frac{1}{|\mathcal{W}^i|} \sum_{w \in \mathcal{W}^i} w$
- 4: $z_{sys}^i := \operatorname{argmin}_{z \in \mathcal{Z}_{sys}^i} \|z - w^i\|_p$
- 5: $\mathcal{V}_{LSR} \leftarrow$ create node z_{sys}^i
- 6: **for each** edge $e = (v_1, v_2) \in \mathcal{E}$ **do**
- 7: find $\mathcal{Z}_{sys}^i, \mathcal{Z}_{sys}^j$ containing v_1, v_2 , respectively
- 8: $\mathcal{E}_{LSR} \leftarrow$ create edge (z_{sys}^i, z_{sys}^j)

return LSR

of disjoint clusters \mathcal{W} , also called *flat* or *partitional* clusters [39], from the stepwise dendrogram M [6] (line 2.2). Points in each cluster \mathcal{W}^i are then assigned a uniform ε^i (line 2.5), i. e. the neighbourhood size from Assumption 2 of each point $z \in \mathcal{W}^i$ is $\varepsilon_z = \varepsilon^i$. We discuss the definition of the ε^i value at the end of this phase. The union of the ε^i -neighbourhoods of the points in \mathcal{W}^i then forms the covered region \mathcal{Z}_{sys}^i (line 2.6). Illustrative examples of covered regions obtained from different values of τ are visualized on the bottom row of Fig. B.6 using various colors. The optimization of τ is discussed in Appendix 12.3. The result of this phase is the set of the identified covered regions $\mathcal{Z}_{sys} = \{\mathcal{Z}_{sys}^i\}$ (line 2.7).

We propose to approximate ε^i as

$$\varepsilon^i = \mu^i + \sigma^i \tag{B.7}$$

where μ^i and σ^i are the mean and the standard deviation of the distances $\|z_j^i - z_k^i\|_p$ among all the training pairs $(z_j^i, z_k^i) \in \mathcal{T}_z$ belonging to the i th cluster. The approximation in (B.7) allows to take into account the cluster density such that denser clusters get lower ε^i . In contrast to our previous work [7], we now enable clusters to have different ε values. We validate the approximation (B.7) in Secs. 9.3.5 and 10.3.1 where we analyze the covered regions identified by the LSR.

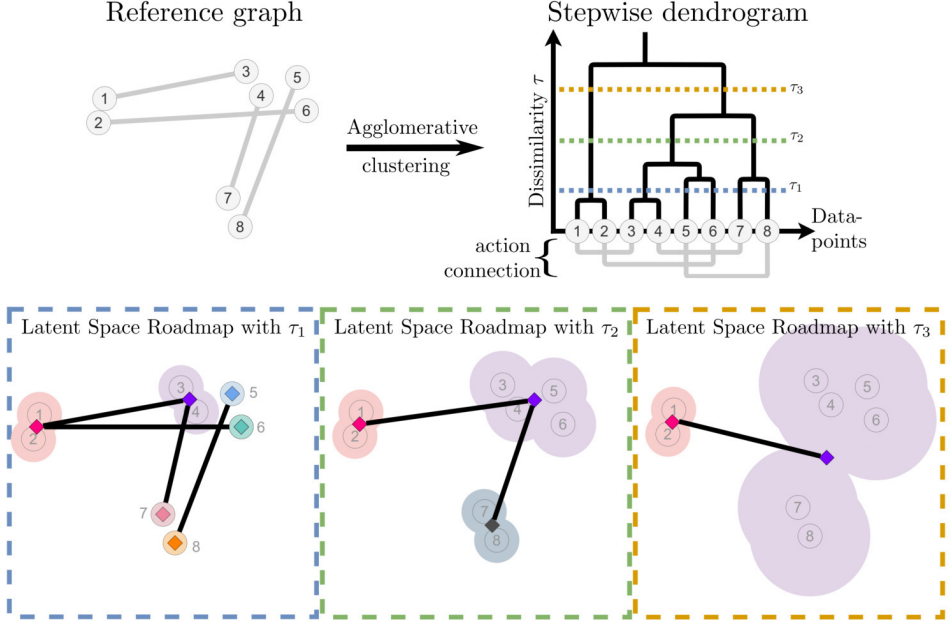


Figure B.6: Illustrative example visualising the LSR building steps and the effect of the clustering threshold τ . The top left shows the reference graph built in Phase 1 of Algorithm 2. The top right part visualizes a dendrogram M obtained from the clustering algorithm in Phase 2. On the bottom row, three examples of LSRs are shown together with the covered regions (marked with various colors) corresponding to different clustering thresholds τ (with $\tau_1 < \tau_2 < \tau_3$) chosen from M .

In Phase 3, we build the LSR = $(\mathcal{V}_{\text{LSR}}, \mathcal{E}_{\text{LSR}})$. We first compute the mean value w^i of all the points in each cluster \mathcal{W}^i (line 3.3). As the mean itself might not be contained in the corresponding path-connected component, we find the equivalence class representative $z_{\text{sys}}^i \in \mathcal{Z}_{\text{sys}}^i$ that is the closest (line 3.4). The found representative then defines a node $v_i \in \mathcal{V}_{\text{LSR}}$ representing the covered region $\mathcal{Z}_{\text{sys}}^i$ (line 3.5). Lastly, we use the set of edges \mathcal{E} in the reference graph built in Phase 1 to infer the transitions $f_z^{i,j}$ between the covered regions identified in Phase 2. We create an edge in LSR if there exists an edge in \mathcal{E} between two vertices in \mathcal{V} that were allocated to different covered regions (lines 3.6 – 3.8). The bottom row of Fig. B.6 shows the final LSRs, obtained with different values of the clustering threshold τ .

Note that, as in the case of the VAE (Sec. 5), no action-specific information u is used in Algorithm 2 but solely the binary variable a indicating the occurrence of an action.

6.2 Optimization of LSR Clustering Threshold τ

The clustering threshold τ , introduced in Phase 2 of Algorithm 2, heavily influences the number and form of the resulting clusters. Since there is no inherent way to prefer one cluster configuration over another, finding its optimal value is a non-trivial problem and subject to ongoing research [40], [41], [42]. However, in our case, since the choice of τ subsequently influences the resulting LSR, we can leverage the information about the latter in order to optimize τ .

As illustrated in Fig. B.6, the number of vertices and edges in LSR_{τ_i} changes with the choice of τ_i . Moreover, the resulting LSRs can have different number of *graph*-connected components. For example, LSR_{τ_1} in Fig. B.6 has 2 graph-connected components, while LSR_{τ_2} and LSR_{τ_3} have only a single one. Ideally, we want to obtain a graph that exhibits both good connectivity which best approximates the true underlying dynamics of the system, and has a limited number of graph-connected component. Intuitively, high number of edges increases the possibility to find latent paths from start to goal state. At the same time, this possibility is decreased when the graph is fragmented into several isolated components, which is why we are also interested in limiting the maximum number of graph-connected components.

While we cannot analyze the clusters themselves, we can evaluate information captured by the LSR that correlates with the performance of the task, i.e., we can assess a graph by the number of edges and graph-connected components it exhibits as discussed above. This induces an objective which we can use to optimize the value of the clustering threshold τ . We formulate it as

$$\psi(\tau, c_{\max}) = \begin{cases} |\mathcal{E}_{\text{LSR}_\tau}| & \text{if } c_{\text{LSR}_\tau} \leq c_{\max}, \\ -\infty & \text{otherwise,} \end{cases} \quad (\text{B.8})$$

where $|\mathcal{E}_{\text{LSR}_\tau}|$ is the cardinality of the set $\mathcal{E}_{\text{LSR}_\tau}$, c_{LSR_τ} represents the number of graph-connected components of the graph LSR_τ induced by τ , and the hyper-parameter c_{\max} represents the upper bound on the number of graph-connected components. The optimal τ in a given interval $[\tau_{\min}, \tau_{\max}]$ can be found by any scalar optimization method. In this work, we use Brent’s optimization method [43] maximizing (B.8):

$$\max_{\tau_{\min} \leq \tau \leq \tau_{\max}} \psi(\tau, c_{\max}). \quad (\text{B.9})$$

This optimization procedure is summarized in Algorithm 3. It takes as an input the encoded training data \mathcal{T}_z , the metric L_p , the search interval where the clustering parameter τ is to be optimized, and the upper bound c_{\max} to compute the optimization objective in (B.8). After initialization of the parameter τ (line 1), for example, by considering the average value of its range, the Brent’s optimization loop is performed (lines 2-5). Firstly, the LSR with the current τ is built according to Algorithm 2 (line 3). Secondly, the optimization objective (B.8) is computed on the obtained LSR_τ (line 4). Thirdly, the parameter τ as well as the bounds τ_{\min} and τ_{\max} are updated according to [43] (line 5). The optimization loop is performed

until the convergence is reached, i. e. , until $|\tau_{\max} - \tau_{\min}|$ is small enough according to [43]. Lastly, the optimal τ^* (line 6) is selected for the final LSR_{τ^*} .

Note that even though Algorithm 3 still needs the selection of the hyperparameter c_{\max} , we show in Sec. 9.3.3 that it is rather robust to the choice of this parameter.

Algorithm 3 LSR input optimization

Require: Dataset \mathcal{T}_z , metric L_p , search interval $[\tau_{\min}, \tau_{\max}]$, c_{\max}

1: $\tau \leftarrow \text{init}(\tau_{\min}, \tau_{\max})$

2: **while** $|\tau_{\max} - \tau_{\min}|$ not small enough **do**

3: $\text{LSR}_{\tau} \leftarrow \text{LSR-building}(\mathcal{T}_z, L_p, \tau)$ [Algorithm 2]

4: $\psi \leftarrow \text{Evaluate}(\text{LSR}_{\tau})$ [Eq. (B.8)]

5: $\tau, \tau_{\min}, \tau_{\max} \leftarrow \text{Brent-update}(\psi)$ [43]

6: $\tau^* \leftarrow \tau$

return LSR_{τ^*}

6.3 Visual plan generation

Given a start and goal observation, a trained VAE model and an LSR, the observations are first encoded by ξ into the VAE’s latent space \mathcal{Z} where their closest nodes in the LSR are found. Next, all shortest paths in the LSR between the identified nodes are retrieved. Finally, the equivalence class representatives of the nodes comprising each of the found shortest path compose the respective latent plan P_z , which is then decoded into the visual plan P_I using ω .

7 Action Proposal Module (APM)

The final component of our framework is the Action Proposal Module (APM) which is used to complement a latent plan, produced by the LSR, with an action plan that can be executed by a suitable framework. The APM allows to generate the action plans from the extracted low-dimensional state representations rather than high-dimensional observations. The action plan P_u corresponding to a latent plan P_z produced by the LSR is generated sequentially: given two distinct consecutive latent states (z_i, z_{i+1}) from P_z , APM predicts an action u_i that achieves the transition $f^{i,i+1}(z_i, u_i) = z_{i+1}$. Such functionality can be realized by any method that is suitable to model the action specifics of the task at hand.

We model the action specifics with a neural network called Action Proposal Network (APN). We deploy a multi layer perceptron and train it in a supervised fashion on the latent *action* pairs obtained from the enlarged dataset \mathcal{T}_z as described below. We validate this choice in Sec 10.4 where we compare it to different alternatives that produce action plans either by exploiting the LSR or by using the observations as inputs rather than extracted low-dimensional states.

The training dataset $\overline{\mathcal{T}}_z$ for the APN is derived from \mathcal{T}_I but preprocessed with the VAE encoder representing the latent mapping ξ . We encode each training *action* pair $(I_1, I_2, \rho = (1, u)) \in \mathcal{T}_I$ into \mathcal{Z} and obtain the parameters μ_i, σ_i of the approximate posterior distributions $q(z|I_i) = N(\mu_i, \sigma_i)$, for $i = 1, 2$. We then sample $2S$ novel points $z_1^s \sim q(z|I_1)$ and $z_2^s \sim q(z|I_2)$ for $s \in \{0, 1, \dots, S\}$. This results in $S + 1$ tuples (μ_1, μ_2, ρ) and $(z_1^s, z_2^s, \rho), 0 \leq s \leq S$, where $\rho = (1, u)$ was omitted from the notation for simplicity. The set of all such low-dimensional tuples forms the APN training dataset $\overline{\mathcal{T}}_z$.

Remark 4. *It is worth remarking the two-fold benefit of this preprocessing step: not only does it reduce the dimensionality of the APN training data but also enables enlarging it with novel points by factor $S + 1$. Note that the latter procedure is not possible with non-probabilistic realizations of ξ .*

8 Assumptions, Applicability and limitations of the method

In this section, we briefly overview our assumptions, describe tasks where our method is applicable, and discuss its limitations. In order for our method to successfully perform a given visual action planning task, the observations contained in the training dataset \mathcal{T}_I should induce the *covered* states (defined in Sec. 3) that are considered in the planning. Furthermore, it is required that sufficiently many transitions among them are observed such that the obtained LSR adequately approximates the true underlying system dynamics. For example, the training datasets \mathcal{T}_I in the box stacking tasks consist of 2500 pairs of states of the system instead of all (i.e., 41616) possible combinations. On the other hand, if the system contains many feasible states, it can be challenging to collect a dataset \mathcal{T}_I that covers sufficiently many states and transitions between them. Even though the performance of the LSR would deteriorate with such incomplete dataset, we do not consider this as the limitation of the method itself as this can be mitigated with online learning approaches, e.g., [44], that dynamically adapt the LSR based on the interaction with the environment.

Given the assumptions on the format of the dataset \mathcal{T}_I introduced in Sec. 4.1, our method is best applicable to visual action planning tasks where feasible states of the system are finite and can be distinguished in \mathcal{T}_I such that meaningful unambiguous actions to transition among them can be defined.

Therefore, our approach does not generalize well to *entirely novel* states of the system not contained in the training set. This is expected, as the model has no prior knowledge about the newly appeared state, such as, for example, an entirely new fold of a T-shirt or a new piece of garment. Such generalization could be achieved by integrating active learning approaches which is indeed an interesting future direction.

We emphasise that the proposed method is not limited by the dimensionality of the system’s states since that is reduced via MM.

9 Simulation results

We experimentally evaluated our method on three different simulated tasks: two versions of a box stacking task (Fig. B.7 left) and a combined rope and box manipulation task (Fig. B.7 right), which we refer to as *rope-box manipulation* task. We considered the initial box stacking task used in our previous work [7] (top left), and a modified one where we made the task of retrieving the underlying state of the system harder. We achieved this by *i*) using more similar box textures which made it more difficult to *separate* the states, and *ii*) by introducing different lighting conditions which made observations containing the same states look more dissimilar. We refer to the original setup as the *normal stacking* task denoted by *ns*, and to the modified one as the *hard stacking* task denoted by *hs*.

In the rope-box manipulation task (Fig. B.7 right), denoted by *rb*, a rope connects two boxes constraining their movement. To challenge the visual action planning, we again introduced different lighting conditions as well as the deformability of the rope.

These three setups enable automatic evaluation of the structure of the latent space \mathcal{Z}_{sys} , the quality of visual plans P_I generated by the LSR and MM, and the quality of action plans P_u predicted by the APN. Moreover, they enable to perform a more thorough ablation studies on the introduced improvements of our framework which were not possible in our earlier version of the LSR [7] since the resulting visual action plans achieved a perfect evaluation score.

All setups were developed with the Unity engine [45] and the resulting images have dimension $256 \times 256 \times 3$. In the stacking tasks, four boxes with different textures that can be stacked in a 3×3 grid (dotted lines in Fig. B.7). A grid cell can be occupied by only one box at a time which can be moved according to the *stacking rules*: i) it can be picked only if there is no other box on top of it, and ii) it can be released only on the ground or on top of another box inside the 3×3 grid. In both versions of the stacking task, the position of each box in a grid cell was generated by introducing $\sim 17\%$ noise along x and y axes which was applied when generating both action and no-action pairs. The action-specific information u , shown in Fig. B.7 left, is a pair $u = (p, r)$ of pick p and release r coordinates in the grid modelled by the row and column indices, i. e. , $p = (p_r, p_c)$ with $p_r, p_c \in \{0, 1, 2\}$, and equivalently for $r = (r_r, r_c)$.

In the rope-box manipulation task, two boxes and a rope can be moved in a 3×3 grid with 4 pillars according to the following manipulation rules: i) a box can only be pushed one cell in the four cardinal directions but not outside the grid, ii) the rope can be lifted over the closest pillar, iii) the rope cannot be stretched over more than two cells, meaning the boxes can never be more than one move apart from being adjacent. In this task, the action-specific information u , shown in Fig. B.7 right, denotes whether the rope is moved over the closest pillar (top) or a box is moved in the grid (bottom) with respective pick p and release r coordinates.

According to the above rules, the training datasets \mathcal{T}_I for stacking tasks contain all possible 288 different grid configurations, i. e. , the specification of which box,

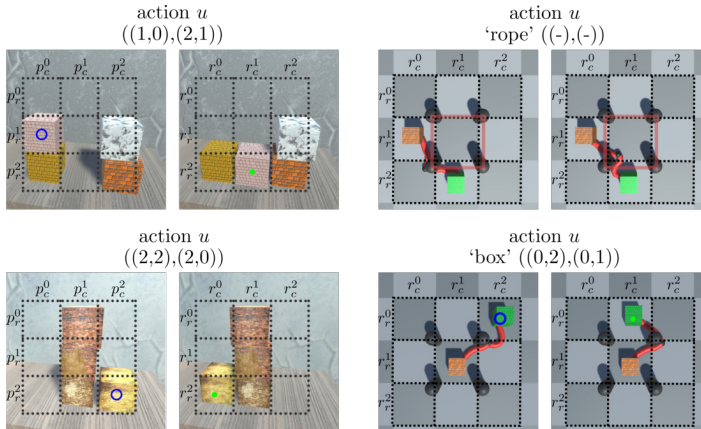


Figure B.7: Examples of actions u in the normal (top) and hard (bottom) box stacking tasks (left) and in the rope-box task (right). The blue circle shows the picking location p , and the green one the release position r . The action ‘rope’ for moving the rope over the closest pillar is shown in top right.

if any, is contained in each cell. In case of the rope-box manipulation task, \mathcal{T}_I contains 157 different grid configurations comprising the position of the rope and boxes. These 288 /157 grid configurations represent the covered states in these tasks. Note that the exact number of underlying states is in general not known. Given a pair of states and the task rules, it is possible to analytically determine whether or not an action is allowed between them. In addition, we can determine the grid configuration associated with an image (i. e. , its underlying state) contained in the produced visual plan P_I using classifiers. These were trained on the decoded images and achieved accuracy greater than 98.8% on a holdout dataset composed of 750 samples for both versions of the stacking task and the rope-box task. All the implementation details can be found on our code repository².

9.1 Experiment Objectives and Implementation Details

Our experiments are designed to answer the following questions:

1. **MM** What is the impact of the action term (B.4) in the augmented loss function (B.6) on the structure of the latent space? How do the respective parameters (e. g. , minimum distance) influence the overall LSR performance? Lastly, how does the VAE framework perform compared to the AE one for modelling the mappings ξ and ω in the MM?

² <https://github.com/visual-action-planning/lsr-v2-code>

2. **LSR** What is the performance of the LSR compared to state of the art solutions like [8] and [9], and what is the influence of the action term (B.4) on it? How do the respective LSR parameters (e.g. , number of components) and the choice of the clustering algorithm impact the overall LSR performance? How good is the LSR approximation of the covered regions?
3. **APM** What is the performance of the APN model?

In this section, we present the implementation details and introduce the notation used to easily refer to the models in consideration. For VAEs (used in MM), each model is annotated by $\text{VAE}_{ld-task-d}$ where ld denotes the dimension of the latent space, $task$ denotes the version of the task and is either ns , hs or rb for the normal stacking task, hard stacking tasks or rope-box manipulation task, respectively. The parameter d indicates whether or not the model was trained with the action loss term (B.4). We use $d = b$ to denote a *baseline* VAE trained with the original VAE objective (B.5), and $d = L_1$ to denote an *action* VAE trained with the loss function (B.6) including the action term (B.4) using metric L_1 . Compared to [7], we consider only L_1 metric in our simulated experiments due to its superior performance over the L_2 and L_∞ metrics established in [7].

All VAE models used a ResNet architecture [46] for the encoder and decoder networks. They were trained for 500 epochs on a training dataset \mathcal{T}_I , composed of 65% action pairs and 35% no-action pairs for stacking tasks, and 50% action pairs and 50% no-action pairs for rope-box manipulation task. For each combination of parameters ld , $task$, and d , we trained 5 VAEs initialized with different random seeds. Same seeds were also used to create training and validations splits of the training dataset. The weight β in (B.5) and (B.6) was gradually increased from 0 to 2 over 400 epochs, while γ was fixed to 100. In this way, models were encouraged to first learn to reconstruct the input images and then to gradually structure the latent space. The minimum distance d_m was dynamically increased every fifth epoch starting from 0 using $\Delta d_m = 0.1$ as described in Sec. 5. The effect of this dynamic parameter increase is shown in Fig. B.5.

For LSR, we denote by $\text{LSR-}L_1$ a graph built using the metric L_1 in Algorithm 2. The parameters τ_{\min} and τ_{\max} in the LSR optimization (B.9) were set to 0 and 3, respectively. Unless otherwise specified, we fixed $ld = 12$ for all tasks. Moreover, the number of graph-components c_{\max} in the optimization of the clustering threshold (B.8) was set to 1 for ns , and 20 for hs and rb . These choices are explained in detail in the following sections. Given an LSR, we evaluated its performance by measuring the quality of the visual plans found between 1000 randomly selected start and goal observations from an unseen holdout set containing 2500 images. To automatically check the validity of the found paths, we used the classifiers on the observations contained in the visual plans to get the respective underlying states. We then defined a checking function (available on the code repository) that, given the states in the paths, determines whether they are allowed or not according to the the stacking or the manipulation rules. In the evaluation of the planning performance we considered the following quantities: *i*) percentage of cases when all

shortest paths from start to goal observations are correct, denoted as % *All*, *ii*) percentage of cases when at least one of the proposed paths is correct, denoted as % *Any*, and *iii*) percentage of correct single transitions in the paths, denoted as % *Trans*. We refer to the % *Any* score in *ii*) as *partial scoring*, and to the combination of scores *i*)-*iii*) as *full scoring*. Mean and standard deviation values are reported over the 5 different random seeds used to train the VAEs.

For APNs, we use the notation $\text{APN}_{id}\text{-task-}d$ analogous to the VAEs. The APN models were trained for 500 epochs on the training dataset $\overline{\mathcal{T}}_z$ obtained following the procedure described in Sec 7 using $S = 1$. Similarly as for LSR, we report the mean and standard deviation of the performance obtained over the 5 random seeds used in the VAE training.

9.2 MM Analysis

In this section, we validate the MM module answering the questions in point 1) of Sec. 9.1. In the first experiment, we investigated the influence of the dynamic parameter d_m on the LSR performance. We then studied the structure of the latent space by analyzing the distance between encodings of different states. Lastly, we compared the LSR performance when modelling MM with an AE framework instead of a VAE.

9.2.1 Influence of dynamic d_m

A key parameter in the action term (B.4) is the minimum distance d_m encouraged among the action pairs. We considered the hard stacking and rope-box manipulation tasks and validated the approach proposed in Sec. 5, which dynamically increases d_m to separate action and no-action pairs (see Fig. B.5). At the end of the training, the approach results in $d_m = 2.3 \pm 0.1$ and $d_m = 2.6 \pm 0.2$ for the hard stacking and rope-box tasks, respectively.

Figure B.8 shows the performance of the LSR using partial scoring on the hard stacking task (blue) and rope-box manipulation task (orange) obtained for the dynamic d_m (solid lines), and a selected number of static d_m parameters (cross markers with dashed lines) ranging from low ($d_m = 0$) to high ($d_m = 100$) values. Among the latter, we included the static $d_m = 11.6$ and $d_m = 6.3$ obtained using our previous approach in [7] on the stacking and the rope-box tasks, respectively. We observed that: *i*) the choice of d_m heavily influences the LSR performance, where same values of d_m can lead to different behavior depending on the task (e. g. , $d_m = 11.6$), *ii*) the dynamic d_m leads to nearly optimal performance regardless of the task compared to the grid searched static d_m . Note that even though there are static d_m values where the performance is higher than in the dynamic case (e. g. , $d_m = 3$ with 93.1% for stacking and $d_m = 9$ with 91.2% for the rope-box task), finding these values a priori without access to ground truth labels is hardly possible.

This approach not only eliminates the need for training the baseline VAEs as in [7] but also reaches a value of d_m that obtains a better separation of covered regions

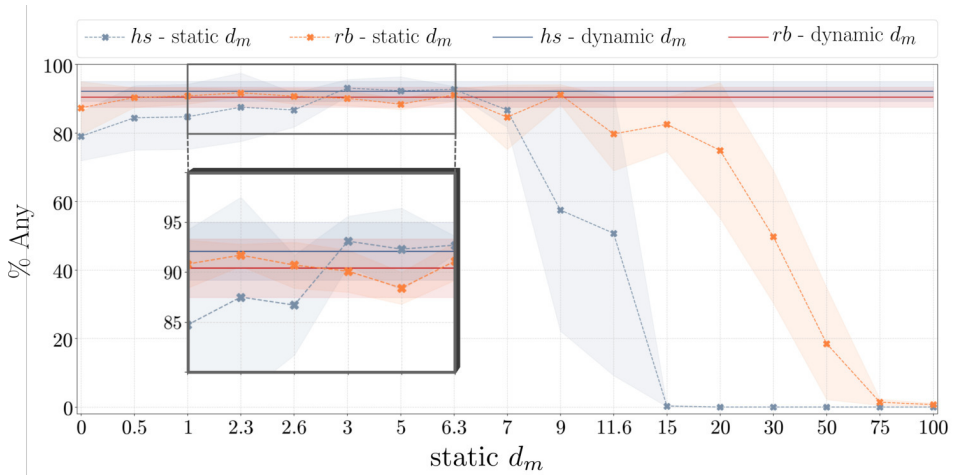


Figure B.8: Comparison of LSR performance using the dynamic d_m (solid lines) and static d_m (cross markers with dashed lines) for the hard stacking (blue) and rope-box manipulation (orange) tasks. Non linear x -axis scale showing the values of d_m is used for better visualization.

\mathcal{Z}_{sys}^i without compromising the optimization of the reconstruction and KL terms. In fact, as discussed in Sec. 5, the reconstruction, KL and action terms in the loss function (B.5) have distinct influences on the latent space structure which can be in contrast to each other. The proposed dynamic increase of d_m results in a lower d_m value than in [7], which in turn yields small distances between the action pair states while still being more beneficial than a simple static $d_m = 0$. Such small distances in the action term are desirable as they do not contradict the KL term. This can explain why the LSRs with higher values of d_m reach worse performance compared to the dynamic one.

9.2.2 Separation of the states

We investigated the effect of the action loss (B.4) on the structure of the latent space by analyzing the separation of the latent points $z \in \mathcal{T}_z$ corresponding to different underlying states of the system. For simplicity, we report only results for the normal stacking task but we observed the same conclusions for the hard stacking and the rope-box manipulation tasks. Recall that images in \mathcal{T}_I containing the same state looked different because of the introduced positioning noise in the stacking tasks (and different lightning conditions in the case of hs as well as the deformability of the rope in rb).

Let \bar{z}_s be the *centroid* for state s defined as the mean point of the training latent samples $\{z_{s,i}\}_i \subset \mathcal{T}_z$ associated with the state s . Let $d_{intra}(z_{s,i}, \bar{z}_s)$ be the *intra-state* distance defined as the distance between the latent sample i associated with

the state s , namely $z_{s,i}$, and the respective centroid \bar{z}_s . Similarly, let $d_{inter}(\bar{z}_s, \bar{z}_p)$ denote the *inter-state* distance between the centroids \bar{z}_s and \bar{z}_p of states s and p , respectively.

Figure B.9 reports the mean values (bold points) and the standard deviations (thin lines) of the inter- (in blue) and intra-state (in orange) distances for each state $s \in \{1, \dots, 288\}$ in the normal stacking task when using the baseline model $\text{VAE}_{12-ns-b}$ (top) and the action model $\text{VAE}_{12-ns-L_1}$ (bottom). In case of the baseline VAE, we observed similar intra-state and inter-state distances. This implies that samples of different states were encoded close together in the latent space which can raise ambiguities when planning. On the contrary, when using $\text{VAE}_{12-ns-L_1}$, we observed that the inter- and intra-state distances approach the values 5 and 0, respectively. These values were imposed with the action term (B.4) as the minimum distance d_m reached 2.6. Therefore, even when there existed no direct link between two samples of different states, and thus the action term for the pair was never activated, the VAE was able to encode them such that the desired distances in the latent space were respected. Similar conclusions also hold for the hard stacking and

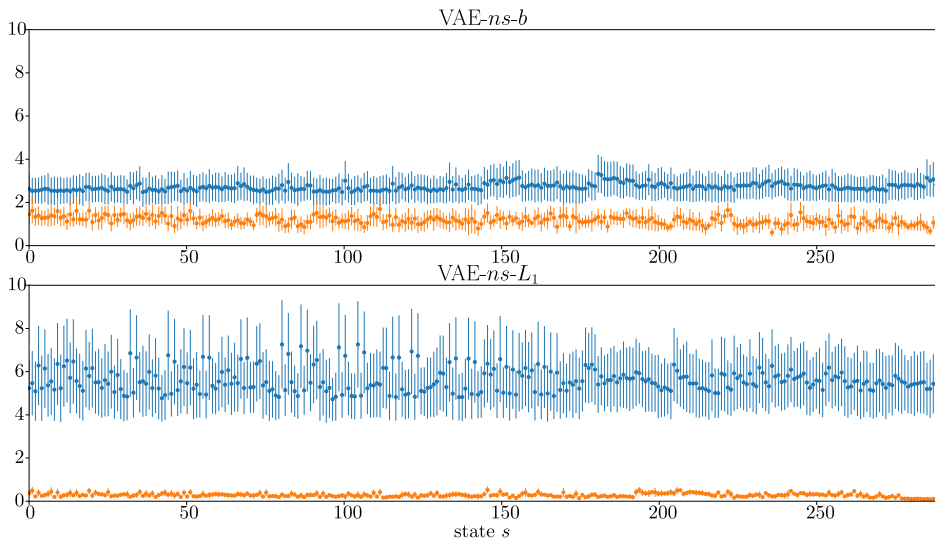


Figure B.9: Mean values (bold points) and standard deviations (thin lines) of inter- (blue) and intra- (orange) state distances for each state calculated using the baseline VAE (top) and the action $\text{VAE}_{12-ns-L_1}$ model (bottom) on normal stacking task.

the rope-box manipulation tasks, whose plots are omitted for the interest of space.

Finally, we analyzed the difference between the minimum inter-state distance and the maximum intra-state distance for each state. The higher the value the better separation of states in the latent space since samples of the same state are

in this case closer to each other than samples of different states. When the latent states were obtained using the baseline VAE_{12-ns-b}, we observed a non-negative distance for 0/288 states with an average value of ≈ -1.2 . This implies that only weak separation occurred in the latent space for samples of different states. On the other hand, when calculated on points encoded with VAE_{12-ns-L₁}, the difference became non-negative for 284/288 states and its mean value increased to ≈ 0.55 , thus achieving almost perfect separation. In the hard stacking task, we similarly found that VAE_{12-hs-b} reached an average difference of -5.86 (being non-negative for 0/288 states), while the action model VAE_{12-hs-L₁} reduced the average difference to -0.04 (being non-negative for 121/288 states). This result demonstrates the difference in the difficulty between the two versions of the box stacking task and highlights the challenges of visual action planning on the harder stacking task where worse separation of states was achieved. For the rope-box manipulation task we obtained, coherently with the box stacking results, an average difference of -2.95 (being non-negative for 37/157 states) with the baseline model, which improved to 0.15 with the action model VAE_{12-rb-L₁} (being non-negative for 100/157 states).

In Appendix12.2, we performed an ablation study on the latent space dimension, justifying the choice $ld = 12$ in our simulations. For each considered latent dimension, we then additionally analyzed the influence of the action loss on the structure of the latent space by calculating *relative contrast* [47, 48] in Appendix12.1, which evaluates the significance of distances among points in the latent space. We observed higher values for action VAEs compared to the baseline VAEs, showing that the action loss leads to higher relative contrast in the latent space.

We conclude that the action term (B.4) and the dynamic d_m contribute to a better structured latent space \mathcal{Z}_{sys} .

9.2.3 VAE compared to AE

VAE framework is only one of the possible models for the MM. We justify this modeling choice by comparing it to the AE framework. Similarly as VAE, an AE model consists of an encoder and a decoder network which are jointly trained to minimize the the Mean Squared Error (MSE) between the original input and its decoded output. In contrast to VAEs, the two networks in AEs do not model a probability distribution. Since the KL divergence in VAE acts as a regularization term, we employed the stable weight-decay Adam optimizer from [49] with default parameters to make the comparison more fair. We denote the model by AE-*b*. Analogously to VAE, the original AE loss was augmented with the action loss (B.4) weighted by the parameter γ , which we denote by AE-*L₁*. Note that *L₁* refers only to the metric in (B.4) and not in the calculation of MSE.

We modelled the AE encoder and decoder networks using the same ResNet [46] architecture as in case of VAEs. We set $ld = 12$, $\gamma = 1000$ and increased the minimum distance d_m dynamically every fifth epoch starting from 0 using $\Delta d_m = 1$, as described in Sec. 5. The LSR was built using the same $\tau_{\min} = 0$ and $\tau_{\max} = 3$ (Algorithm 3).

Table B.2 shows the LSR performance using partial scoring on all simulated tasks when MM was modelled as an AE (top two rows) and as a VAE (bottom row). Not only we observed a superior performance of VAE compared to the AE but once again the effectiveness of the action term (B.4) on all the tasks as it increased the average AE performance from 0.1% to 36.3% for ns , from 0.1% to 33.6% for hs , and 0.1% to 9.8% for rb . *This comparison shows that the probabilistic modeling adopted by VAEs resulted in a latent space that is more adequate for visual action planning with respect to the considered AEs.*

Model	ns [%]	hs [%]	rb [%]
AE- b +LSR- L_1	0.1 ± 0.0	0.0 ± 0.0	0.1 ± 0.1
AE- L_1 +LSR- L_1	36.3 ± 26.9	33.6 ± 10.3	9.8 ± 5.4
VAE- L_1 +LSR- L_1	100.0 ± 0	92.1 ± 2.9	90.4 ± 2.9

Table B.2: Comparison of the LSR performance using partial scoring when modelling MM with an AE (top two rows) and a VAE (bottom row) framework on all the simulated tasks. Best results in bold.

9.3 LSR Analysis

In this section, we analyze the LSR performance by answering the questions stated in point 2) of Sec. 9.1. Firstly, we compared the LSR performance to the method in [8] and one inspired by [9]. Secondly, we investigated the influence of the action term (B.4) on the LSR performance. Thirdly, we investigated the influence of the upper bound on the number of connected components c_{\max} used in (B.8). Next, we performed an extensive comparison of the LSR algorithm using different clustering algorithms in Phase 2 of Algorithm 2. Finally, we analyzed the covered regions determined by the LSR.

9.3.1 LSR comparison

We compared the performance of the LSR on all simulated tasks with two benchmark methods introduced below. In all the experiments, we considered the baseline models VAE $_{12-b}$ and the action VAE $_{12-L_1}$ trained with the action term (B.4).

We compared our method with Semi-Parametric Topological Memory (SPTM) framework presented in [8] and an MPC-based approach inspired by [9].

In SPTM, we connected action pairs (treated as one-step trajectories) and no-action pairs (considered temporarily close) in the latent memory graph. As in [8], we added N_{sc} more *shortcut* edges connecting the encodings that are considered closest by the retrieval network to the memory graph. In the localization step, we used the median of $k = 5$ nearest neighbours of the nodes in the memory graph as recommended in [8]. To select the waypoint, we performed a grid search over $s_{\text{reach}} \in \{0.75, 0.9, 0.95\}$ and chose $s_{\text{reach}} = 0.95$. We also performed

a grid search over $N_{sc} \in \{0, 2 \cdot 10^2, 1 \cdot 10^4, 1 \cdot 10^5, 1 \cdot 10^6, 1.5 \cdot 10^6, 2 \cdot 10^6\}$ and used the values $N_{sc} = 1.0 \cdot 10^6, 1.5 \cdot 10^6, 2.0 \cdot 10^6$ for *ns*, *hs* and *rb*, respectively. We used high number of shortcuts compared to $N_{sc} = 2 \cdot 10^2$ in [8] because we only had access to one-step trajectories instead of full roll-outs. Using low number of shortcuts resulted in a memory graph consisting of large amount of *disconnected* components which impeded planning. For example, in hard stacking task using $N_{sc} = 2 \cdot 10^2$ yielded a graph with 2243 connected components which led to almost zero correct transitions over the 1000 test paths. A higher number of shortcuts instead improved the connectivity of the graph and thus its planning capabilities.

The MPC-inspired baseline is composed of a learned transition model $f_t(\cdot)$ and a learned action validation model $f_a(\cdot)$, both taking the current latent state z_1 and the applied action u as inputs. The transition model then predicts the next state $z_2 = f_t(z_1, u)$, while the validation model $f_a(z_1, u)$ predicts whether the given action u was allowed or not.

These models are used in a MPC-style approach, where first a search tree is constructed for a given start state z_1 by iterating over all allowed action using $f_a(z_1, u)$ with $u \in \mathcal{U}$ and predicting the consecutive states with the transition model $f_t(\cdot)$. The search is performed at each time step and until the search tree has reached a specified horizon N . Lastly, the path in the built tree leading to the state closest to the goal using L_1 distance is selected and the first action in the sequence is applied. This procedure is repeated until all proposed actions lead further from the goal. In our case, the resulting state and action sequence is decoded into a visual action plan and evaluated in the same way as the LSR.

We implemented f_t and f_a as a three layer MLP-regressor and MLP-classifier, respectively, with 100 hidden units. For a fair comparison, we trained f_t and f_a using training encodings \mathcal{T}_z from the same MM that was used for building the LSR. As \mathcal{T}_z only includes allowed actions, we augmented the training data for $f_a(\cdot)$ with an equal amount of negative examples by randomly sampling $u \in \mathcal{U}$. We used horizon $N = 4$. The trained f_t models achieved R^2 coefficient of determination [50] of 0.96, 0.96, and 0.88 (highest 1) for the normal, hard stacking and rope-box datasets, respectively. The $f_a(\cdot)$ model was evaluated on 1000 novel states and by applying all possible actions on each state. It achieved an accuracy score of 88.5 ± 1.8 , 97.3 ± 0.2 , and 87.4 ± 0.8 for the normal, hard stacking and rope-box datasets, respectively. Note that the normal and hard stacking tasks has exactly 48 unique actions with $\approx 9.4\%$ of them being allowed on average. The rope-box task on the other hand has 25 unique actions with an average of $\approx 17.1\%$ being allowed per state.

Table B.3 shows the result of our method (VAE- L_1 + LSR- L_1), the SPTM framework and the MPC-based approach (VAE- L_1 + MPC) evaluated on the full scoring on the normal box stacking (top), hard box stacking (middle), and rope-box manipulation task (bottom). We observed that the proposed approach (VAE- L_1 + LSR- L_1) significantly outperformed the considered benchmark methods. This can be explained by the fact that SPTM- and MPC-based methods are more suited for tasks where the provided data consists of rolled out *trajectories* in which small

state changes are recorded in consecutive states. In contrast, as discussed in Sec. 8, our method is best applicable when actions lead to distinguishable different observations. This allows to consider only pairs of observations as input dataset instead of requiring entire trajectories. Moreover, a core difference between our approach and SPTM is that we do not assume that each observation maps into a unique underlying state, but rather, as described in Sec. 4, we structure and cluster observations in such a way that observations associated with the same underlying state are grouped together. We reiterate that this approach is best suited for tasks with finite and distinguishable states, which differ from continuous RL setting used by SPTM.

Task	Model	% All	% Any	% Trans.
<i>ns</i>	VAE- L_1 + MPC	2.3 ± 0.3	2.3 ± 0.3	69.3 ± 1.0
	SPTM [8]	0.2 ± 0.1	0.5 ± 0.3	51.9 ± 1.4
	VAE- b + LSR- L_1	2.5 ± 0.5	4.1 ± 1.0	59.7 ± 4.9
	VAE- L_1 + LSR- L_1	100.0 ± 0	100.0 ± 0	100.0 ± 0
<i>hs</i>	VAE- L_1 + MPC	2.1 ± 0.4	2.1 ± 0.4	76.8 ± 0.3
	SPTM [8]	0.0 ± 0.0	0.0 ± 0.0	23.6 ± 0.7
	VAE- b + LSR- L_1	0.2 ± 0.1	0.2 ± 0.1	38.0 ± 2.0
	VAE- L_1 + LSR- L_1	90.9 ± 3.5	92.1 ± 2.9	95.8 ± 1.3
<i>rb</i>	VAE- L_1 + MPC	6.2 ± 0.5	6.2 ± 0.5	73.8 ± 0.8
	SPTM [8]	0.0 ± 0.0	0.4 ± 0.3	25.2 ± 9.7
	VAE- b + LSR- L_1	0.2 ± 0.1	0.2 ± 0.1	0.2 ± 0.1
	VAE- L_1 + LSR- L_1	89.7 ± 3.7	90.4 ± 2.9	96.2 ± 1.5

Table B.3: Planning performance using full scoring for the normal (top) and hard (middle) box stacking tasks and rope-box manipulation task (bottom) using MPC and SPTM [8] methods, baseline VAE- b and action VAE- L_1 . Best results in bold.

9.3.2 Influence of the action term

We investigated how the LSR performance is affected by the action term (B.4) by comparing it to the variant where MM was trained without it (VAE- b + LSR- L_1). The results on the full scoring for all the tasks are shown in Table B.3. We observed deteriorated LSR performance when using baselines VAE- b compared to the action VAEs regardless the task. This indicates that VAEs- b were not able to separate states in \mathcal{Z}_{sys} . We again conclude that the action term (B.4) needs to be included in the VAE loss function (B.6) in order to obtain distinct covered regions \mathcal{Z}_{sys}^i . In addition, the results underpin the different level of difficulty of the tasks as indicated by the drop in the LSR performance on *hs* and *rb* compared to *ns* using the action VAE- L_1 .

In summary, this simulation campaign demonstrates the effectiveness of the LSR on all the considered simulated tasks involving both rigid and deformable objects

compared to existing solutions, as well as supports the integration of the action term in the VAE loss function.

9.3.3 Influence of the maximum number of connected components

The optimization method described in Sec. 6.2 requires setting an upper bound on the number of graph-connected components c_{\max} of the LSR. Table B.4 shows how different choices of upper bounds influence the LSR performance on all simulated tasks.

c_{\max}	ns [%]	hs [%]	rb [%]
1	100.0 ± 0.0	65.3 ± 24.6	4.5 ± 5.6
5	99.5 ± 0.4	88.6 ± 5.4	55.8 ± 28.8
10	99.0 ± 0.3	91.5 ± 3.8	80.4 ± 10.6
20	97.5 ± 0.5	92.1 ± 2.9	90.4 ± 2.9
50	91.3 ± 1.1	88.2 ± 2.0	89.4 ± 1.9
100	80.0 ± 1.4	77.9 ± 2.1	76.0 ± 2.8

Table B.4: LSR performance on all simulated tasks for different c_{\max} values. Best results in bold.

We observed that the results are rather robust with respect to the c_{\max} value. For all tasks, the performance dropped for a very high c_{\max} , such as $c_{\max} = 100$, while in the hard stacking task and especially in the rope-box manipulation task, we additionally observed a drop for a very low c_{\max} , such as $c_{\max} = 1$. This behavior can be explained by the fact that the lower the c_{\max} the more the system is sensitive to outliers, while the higher the c_{\max} the greater the possibility that the graph is disconnected which potentially compromises its planning capabilities. For example, in the hard stacking task, outliers arise from different lightning conditions, while in the rope-box manipulation task they arise from the deformability of the rope. In contrast, no outliers exist in the normal stacking task which is why a single connected component is sufficient for the LSR to perform perfectly. For all further evaluation, we set $c_{\max} = 1$ for ns and $c_{\max} = 20$ for hs and rb .

This result demonstrates the robustness of the approach with respect to c_{\max} as well as justifies the choices of the c_{\max} values in the rest of simulations.

9.3.4 Comparing different clustering methods for Phase 2

We showcase the effect of the outer optimization loop described in Algorithm 3 on several different clustering methods used in Phase 2 in Algorithm 2 on the hard stacking task. We considered *Epsilon clustering* used in our earlier work [7], *Mean-shift* [51], *OPTICS* [52], *Linkage* (single, complete and average) [53] and *HDBSCAN* [54] algorithms. We provide a summary of the considered algorithms in Appendix 12.3. The performance of the considered clustering methods (except for HDBSCAN) depends on a single input scalar parameter that is hard to tune.

However, as described in Sec. 6.2, we are able to optimize it by maximizing the objective in (B.8).

Table B.5 reports the LSR performance with different clustering algorithms when performing grid search to determine their input scalar parameters (left) and when using our automatic optimization (right). Partial scoring using $\text{VAE}_{12-hs-L_1}$ is shown. Note that the grid search was only possible in this problem setting as the ground truth can be retrieved from the trained classifiers but it is not generally applicable. Firstly, the results show that average-linkage, used for our LSR in Sec. 6.1, together with our automatic input parameter optimization outperformed the other alternatives. The results of the grid search show that the automatic criteria for identifying different cluster densities, adopted by OPTICS and HDBSCAN, did not effectively retrieve the underlying covered regions. Meanshift performed better but its approximation of spherical clusters did not lead to the optimal solution. Similar performance to Meanshift was obtained with single- and complete-linkage algorithms showing that the respective distance functions are not either suited for identifying covered regions. The same applies for the epsilon clustering.

Concerning the optimization results, they highlight the effectiveness of the optimization procedure in Algorithm 3 as they are comparable to the ones obtained with the grid search for all clustering methods. Note that grid search led to a slightly lower performance than the optimization for meanshift, complete-linkage and average-linkage. In these cases, the grid was not fine enough which points out the difficulty of tuning the respective parameters.

This investigation demonstrates the effectiveness of our proposed optimization loop and shows that the average-linkage clustering algorithm led to the best LSR performance among considered alternatives for the hard box stacking task.

Clust. method	Grid Search [%]	Optimization [%]
Epsilon [7]	83.5 ± 4.8	65.8 ± 12.2
Meanshift	78.2 ± 3.3	80.2 ± 5.9
OPTICS	44.3 ± 8.7	40.8 ± 6.1
HDBSCAN	16.1 ± 5.7	-
Single-linkage	79.3 ± 8.8	65.8 ± 12.2
Complete-linkage	79.1 ± 6.4	81.4 ± 4.8
Average-linkage	91.1 ± 2.5	92.1 ± 2.9

Table B.5: Comparison of the LSR performance for different clustering algorithms for the hard box stacking task. Partial scoring is reported when applying grid search (left column) and when using the optimization in Algorithm 3 (right column). Best results in bold.

9.3.5 Covered regions using LSR

To show that the LSR captures the structure of the system, we checked if observations corresponding to true underlying states of the system, that have not been seen during training, are properly recognized as covered. Then, we checked if observations from the datasets of the remaining simulated tasks as well as from the 3D Shapes dataset [55] are marked as uncovered since they correspond to out-of-distribution observations. The covered regions \mathcal{Z}_{sys}^i were computed using the epsilon approximation in (B.7).

Table B.6 reports the results of the classification of covered states obtained by the models trained on normal (first row) and hard (second row) box stacking tasks and rope-box manipulation task (third row). Holdout datasets for each simulated task were used. For the normal stacking task, results show that the LSR almost perfectly recognized all the covered states (*ns* column) with the average recognition equal to 99.5%, while it properly recognized on average 4694/5000 samples (93.9% - *hs* column) hard version. An almost perfect average recognition was also obtained on the rope-box manipulation task (99.6% - *rb* column). For out-of-distribution observations, the lower the percentage the better the classification. Table B.6 shows that the models trained on *ns* (first row, columns *hs*, *rb*, 3D Shapes) and *hs* (second row, columns *ns*, *rb*, 3D Shapes) were able to perfectly identify all *non*-covered states, while worse performance was observed for the rope-box models which misclassified $\approx 10\%$ of the uncovered datasets (third row, columns *ns*, *hs*, 3D Shapes). This decrease in performance could be explained by the fact that capturing the state of a deformable object is much more challenging than rigid objects.

We conclude that LSR provides a good approximation of the global structure of the system as it correctly classified most of the observations representing possible system states as covered, and out-of-distribution observations as not covered.

	<i>ns</i> [%]	<i>hs</i> [%]	<i>rb</i> [%]	3D Sh. [%]
<i>ns</i>	99.47 ± 0.27	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
<i>hs</i>	0.0 ± 0.0	93.71 ± 0.61	0.0 ± 0.0	0.0 ± 0.0
<i>rb</i>	9.48 ± 7.45	13.5 ± 8.57	99.6 ± 0.1	9.72 ± 8.38

Table B.6: Classification of covered states for the normal (first row) and hard (second row) box stacking models and rope-box models (third row) when using as inputs novel images from the tasks (*ns*, *hs* and *rb* columns) and the 3D Shapes (3D Sh. column) datasets.

9.4 APM Analysis

We evaluated the accuracy of action predictions obtained by APN- L_1 on an unseen holdout set consisting of 1611, 1590 and 948 action pairs for the normal stacking, hard stacking and rope-box manipulation tasks, respectively. As the predicted

actions can be binary classified as either true or false, we calculated the percentage of the correct proposals for picking and releasing, as well as the percentage of pairs where both pick and release proposals were correct. For rope-box task, we additionally calculated the percentage of the correct proposal for either rope or box action. We evaluated all the models on 5 different random seeds. For both stacking versions, all the models performed with accuracy 99% or higher, while rope-box models achieved $\approx 96\%$. This is because the box stacking task results in an 18-class classification problem for action prediction which is simple enough to be learned from any of the VAEs, while the classification task in the rope-box is slightly more challenging due to the required extra prediction whether to move a rope or a box.

10 Folding Experiments

In this section, we validate the proposed approach on a real world experiment involving manipulation of deformable objects, namely folding a T-shirt. As opposed to the simulated tasks, the true underlying states were in this case unknown and it was therefore not possible to define an automatic verification of the correctness of a given visual action plan.

The folding task setup is depicted in Fig. B.12 (middle). We used a Baxter robot equipped with a Primesense RGB-D camera mounted on its torso to fold a T-shirt in different ways. The execution videos of all the performed experiments and respective visual action plans can be found on the project website. A summary of the experiments can also be found in the accompanying video. For this task, we collected a dataset \mathcal{T}_I containing 1283 training tuples. Each tuple consists of two images of size $256 \times 256 \times 3$, and action specific information u defined as $u = (p, r, h)$ where $p = (p_r, p_c)$ are the picking coordinates, $r = (r_r, r_c)$ the releasing coordinates and h picking height. An example of an action and a no-action pair is shown in Fig. B.4. The values $p_r, p_c, r_r, r_c \in \{0, \dots, 255\}$ correspond to image coordinates, while $h \in \{0, 1\}$ is either the height of the table or a value measured from the RGB-D camera to pick up only the top layer of the shirt. Note that the separation of stacked clothing layers is a challenging task and active research area on its own [56] and leads to decreased performance when it is necessary to perform it, as shown in Sec. 10.5.2. The dataset \mathcal{T}_I was collected by manually selecting pick and release points on images showing a given T-shirt configuration, and recording the corresponding action and following configuration. No-action pairs, representing $\approx 37\%$ of training tuples in \mathcal{T}_I , were generated by slightly perturbing the cloth appearance.

10.1 Experiment Objectives and Implementation Details

The experiments on the real robot were designed to answer the following questions:

1. **MM** Does the action loss term (B.4) improve the structure of the latent space for the folding task?
2. **LSR** How good is the approximation of the covered regions provided by the LSR for a real world dataset?
3. **APM** How does the APN perform in comparison to alternative implementations of the APM?
4. **System** How does the real system perform and how does it compare to our earlier work [7]? What is the performance on a folding that involves picking the top layer of the shirt?

Following the notations introduced in Sec. 9.1, we denote by VAE_{ld-f-d} a VAE with ld -dimensional latent space, where f stands for the folding task and d indicates whether or not the model was trained with the action loss (B.4). We use $d = b$ for the *baseline* VAEs which were trained with the original training objective (B.5). We use $d = L_p$ for the *action* VAEs trained with the objective (B.6) containing the action term (B.4) using metric L_p for $p \in \{1, 2, \infty\}$. We modelled VAEs with the same ResNet architecture and same hyperparameters β , γ and d_m as in the box stacking task introduced in Sec. 9 but increased the latent space dimension to $ld = 16$. We refer the reader to the code repository² for implementation details.

For the LSR, we denote by $\text{LSR-}L_p$ a graph obtained by using metric L_p in Algorithm 2. We set the upper bound c_{\max} in (B.8) to 5, and the search interval boundaries τ_{\min} and τ_{\max} in Algorithm 3 to 0 and 3.5, respectively.

The performance of the APMs and the evaluation of the system was based on the VAE_{16-f-L_1} realization of the MM. We therefore performed the experiments using APN_{16-f-L_1} which was trained on latent action pairs $\overline{\mathcal{T}}_z$ extracted by the latent mapping ξ of VAE_{16-f-L_1} . We trained 5 models for 500 epochs using different random seeds as in case of VAEs, and used 15% of the training dataset as a validation split to extract the best performing model for the evaluation.

We compared the performance of our system S-OUR consisting of VAE_{16-f-L_1} , $\text{LSR-}L_1$ and APN_{16-f-L_1} with the systems S- L_1 , S- L_2 and S- L_∞ introduced in [7] on the same 5 folding tasks. The start configuration was the fully unfolded shirt shown in Fig. B.10 on the left, while the goal configurations are shown on the right. The latter are of increasing complexity requiring a minimum of 2, 2, 3, 3, and 4 folding steps for folds 1-5, respectively.

Each fold was repeated 5 times and scored in the same way as in [7]. In particular, we scored the *system performance* where a folding was considered successful if the system was able to fold the T-shirt into the desired goal configuration. As the state space of the T-shirt is high-dimensional, there exists no objective measure that would evaluate the success of the fold automatically. Therefore, the evaluation of the full folding procedure was manually done by a human (one of the authors) but all execution videos of all folds and repetitions can be found on the project website. We additionally evaluated the percentage of successful *transitions of the system*.

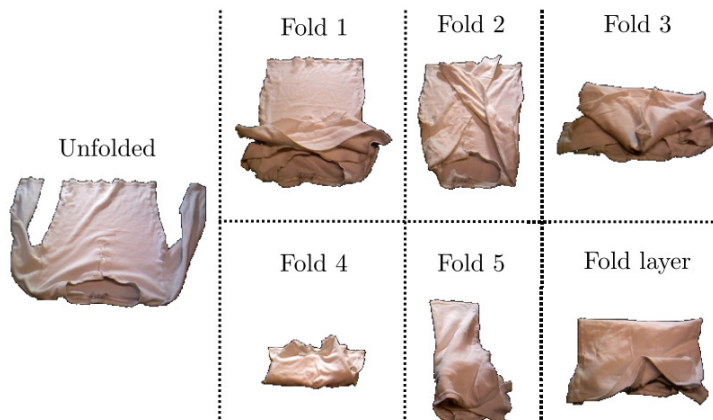


Figure B.10: Start state (right) followed by 5 different goal configurations for the folding task [7]. The lower right configuration requires to pick a layer on top of the T-shirt.

A transition was considered successful if the respective folding step was executed correctly. Lastly, we evaluated the quality of the generated visual plans P_I and the generated action plans P_u . We considered a visual (action) plan successful if all the intermediate states (actions) were correct. Even for a correctly generated visual action plan, the open loop execution is not robust enough for a real robot system. We therefore added a re-planning step after each action completion as shown in Fig. B.12. This accounts, as instance, for potential execution uncertainties, inaccuracies in grasping or in the positioning phases of pick-and-place operations which led to observations different from the ones planned in P_I . Note that after each action execution, the current observation of the cloth was considered as a new start observation, and a new visual action plan was produced until the goal observation is reached or the task is terminated. Such re-planning setup was used for all folding experiments. As the goal configuration does not allude to how the sleeves should be folded, the LSR suggests multiple latent plans. A subset of the corresponding visual action plans is shown on the left of Fig. B.12. If multiple plans were generated, a human operator selected one to execute. After the first execution, the ambiguity arising from the sleeve folding was removed and the re-planning generated a single plan, shown in the right.

To deal with the sparse nature of the collected dataset, if no path was found from the start to the goal node, the planning was repeated using the closest nodes to the current start and/or goal nodes in the latent space. This procedure was repeated until a path was found.

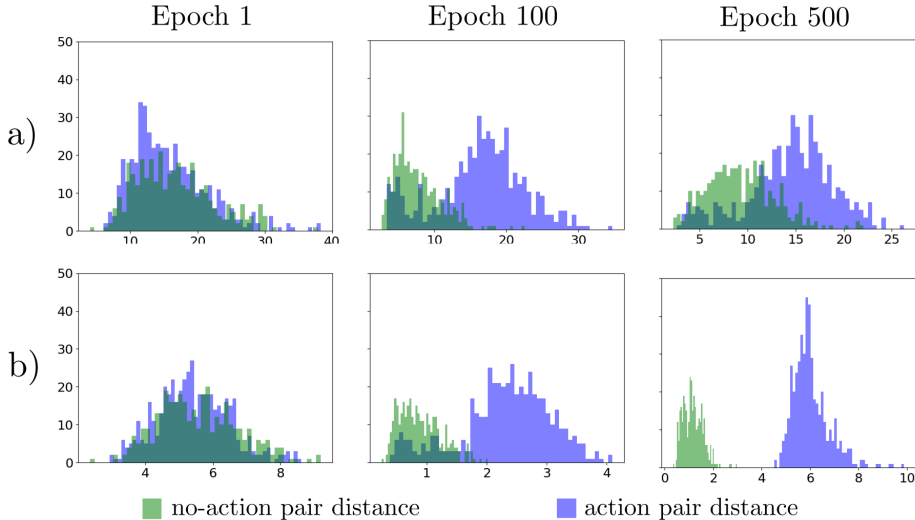


Figure B.11: Histograms of action (in blue) and no-action (in green) pair distances at different training epochs (1, 100 and 500 from the left, respectively) for the folding task. Results obtained with baseline (top, a)) and action (bottom, b)) models are shown.

10.2 MM Analysis

Analogously to the box stacking task in Sec. 9.2, we answered question 1) by analyzing the relative contrast (reported in Appendix 12.1) as well as by evaluating the separation of action and no-action pairs during the training.

10.2.1 Influence of dynamic d_m

We investigated the influence of the dynamic increase of d_m in the action term (B.4) on the structure of the latent space. Figure B.11 shows the histogram of action (in blue) and no-action (in green) pair distances calculated at different epochs during training using VAE₁₆- f - b (top row) and VAE₁₆- f - L_1 (bottom row). We observed that the separation was complete in case of action VAEs but was not achieved with the baseline VAEs. To precisely quantify the amount of overlap between action and no-action pairs, we calculated the difference between the minimum action-pair distance and maximum no-action pair distance on the training dataset. Therefore, a positive value implies that action pairs were successfully separated from the no-action pairs. For VAE₁₆- f - b (top row), the difference evaluated to -31.8 , -19.2 , and -19.4 for epoch 1, 100, and 500, respectively, while it was improved to -6.3 , -1.6 , and 1.5 in case of the action VAE₁₆- f - L_1 (bottom row). *This shows that the*

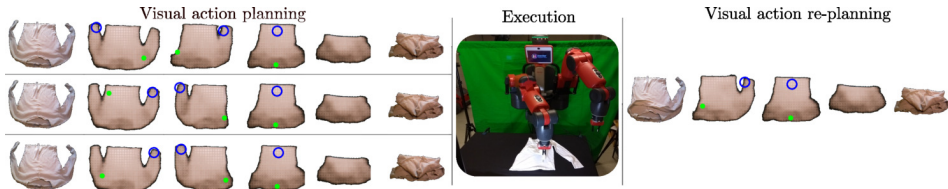


Figure B.12: Execution of the folding task with re-planning. On the left, a set of initial visual action plans reaching the goal state is proposed. After the first execution, only one viable visual action plan remains.

dynamic selection of d_m successfully separated the actions and no-action pairs also for the folding task.

10.3 LSR Analysis

Similarly to the simulated tasks, we exploited the LSR to investigate the covered regions of the latent space \mathcal{Z} , thus answering question 2) listed in Sec. 10.1. Note that in Sec. 10.5, the LSR was also employed to perform the folding task with the real robotic system.

10.3.1 Covered regions using LSR

We used $\text{VAE}_{16} - f - L_1$ model and reproduced the experiment from Sec. 9.3.5, where we measured the accuracy of various novel observations being recognized as covered. We inputted 224 novel observations that correspond to possible states of the system not used during training, as well as 5000 out-of-distribution samples from each of the three datasets of the simulated tasks and the standard 3D Shapes dataset. We observed that the LSR achieved good recognition performance even in the folding task. More precisely, on average 213/224 samples representing true states of the system were correctly recognized as covered, resulting in $95 \pm 2.4\%$ accuracy averaged over the 5 different random seeds. For the four out-of-distribution datasets, all samples were correctly recognized as not covered.

This analysis illustrates the effectiveness of the LSR in capturing the covered regions of the latent space.

10.4 APM Comparison

In this section we validate the choice of the APM by comparing it to several possible alternatives.

The Action Proposal Network, described in Sec. 7, was built upon the one introduced in [7] to which we added dropout regularization layers. The APN receives as inputs latent action pairs contained in a latent plan found by the LSR, and outputs

Method	X Pick	Y Pick	X Release
e-APN [7]	144.1 ± 52.2	52.8 ± 18.3	317.2 ± 143.3
C-APN	498.0 ± 63.8	47.4 ± 7.7	818.8 ± 121.9
R-APN	697.2 ± 345.1	246.2 ± 174.9	792.4 ± 388.8
AAB	113.0	22.4	201.4
APN (Ours)	82.6 ± 22.9	29.3 ± 2.2	270.6 ± 158.2

Method	Y Release	Height	Total
e-APN [7]	159.9 ± 17.4	0.0 ± 0.0	674.0 ± 147.6
C-APN	226.5 ± 92.5	0.0 ± 0.0	1590.8 ± 155.0
R-APN	268.9 ± 157.0	0.0 ± 0.0	2004.6 ± 908.2
AAB	194.7	0.0	531.5
APN (Ours)	71.8 ± 15.0	0.0 ± 0.0	454.3 ± 153.8

Table B.7: Comparison of MSE achieved with different realizations of the Action Proposal Modules. Best results in bold.

the predicted action specifics. We refer to the *earlier* version in [7] as *e-APN* and to the current version APN_{16-f-L_1} as *APN*. We compared the performance of APN to e-APN as well as several alternatives introduced below.

Action Averaging Baseline (AAB) Firstly, we investigated whether the action predictions can be retrieved directly from the LSR instead of a separate module. The basic idea is to use the latent action pairs in the training dataset to calculate the average action specifics associated with each edge in the LSR. Let $\mathcal{E}_{s_{ys}}^{ij} = \{(z_1, z_2) \in \mathcal{E} | z_1 \in \mathcal{Z}_{s_{ys}}^i, z_2 \in \mathcal{Z}_{s_{ys}}^j\}$ be the set of edges from the reference graph \mathcal{E} connecting covered regions $\mathcal{Z}_{s_{ys}}^i$ and $\mathcal{Z}_{s_{ys}}^j$ (Algorithm 2). We parameterized each edge $e_{\text{LSR}}^{ij} = (z_{\text{LSR}}^i, z_{\text{LSR}}^j) \in \mathcal{E}_{\text{LSR}}$ with the action u_{LSR}^{ij} obtained by averaging actions corresponding to the edges in $\mathcal{E}_{s_{ys}}^{ij}$

$$u_{\text{LSR}}^{ij} = \frac{1}{|\mathcal{E}_{s_{ys}}^{ij}|} \sum_{(z_1, z_2) \in \mathcal{E}_{s_{ys}}^{ij}} u^{z_1 z_2} \quad (\text{B.10})$$

where $u^{z_1 z_2}$ is the action specification associated with the action pair (z_1, z_2) in the training dataset \mathcal{T}_z . The parametrization (B.10) yields the action plan associated with a path P_z .

Secondly, we investigated how the use of the latent encodings as inputs to the APM influences the LSR performance. We compared APN-d with two distinct versions of APMs that use images as inputs.

C-APN is a neural network that uses a convolutional encoder followed by the APN. The encoder in C-APN was trained using only MSE loss. During the inference, the observations given to C-APN as input are obtained by decoding the latent plan found by the LSR with the observation generator ω .

R-APN is an extension of C-APN that uses a ResNet encoder identical to the VAE encoder.

Detailed architectures of all the models can be found in our code repository. The training details for APN and APN-d are described in Sec. 10.1. For C-APN-d and R-APN-d, we similarly trained 5 models using different random seeds but on a training dataset \overline{T}_I obtained by decoding \overline{T}_z with the observation generator ω of VAE₁₆- f - L_1 . This is because the visual plans, given to C-APN-d and R-APN-d, are produced by decoding the latent plans with ω . Moreover, C-APN-d and R-APN-d were trained for 1000 epochs to ensure the convergence of the initialized encoders. Note that we can only obtain one AAB model for a chosen VAE as AAB is defined by the LSR.

We evaluated the performance of all the models on a holdout dataset consisting of 41 action pairs. Given a holdout action pair, we calculated the mean squared error (MSE) between the predicted and the ground truth action specifics. We report the mean and standard deviation of the obtained MSE calculated across the 5 random seeds (except for AAB). The results are shown in Table B.7 where we separately report the error obtained on picking and releasing as well as the total model error. Firstly, we observed that the added regularization layer positively affected the result as APN achieved lower error than our earlier version e-APN [7]. Secondly, APN significantly outperformed both C-APN and R-APN. Using the latent encodings as inputs also significantly decreased the size of the models and reduces the computational power needed for their training. Lastly, our APN also on average outperformed AAB with respect to the total model error. Although the enhancement compared to the AAB was not as significant as for the other models, APN is beneficial since it is less prone to averaging errors obtained from the LSR and can be easily adapted to any realization of action specifics. Moreover, a neural network realization of the APM potentially allows more accurate modeling of more complex action specifics. *In summary, using a separate neural network to predict action specifics from latent representations led to a lower prediction error and can be easily adapted to different types of actions.*

10.5 System Analysis

We benchmarked our method against our earlier method in [7] on the same T-shirt folding task, and additionally measured the performance on a more challenging fold involving picking a layer of the cloth on top of another layer.

10.5.1 Folding performance and comparison with [7]

We performed each fold 5 times per configuration using the goal configuration shown in Fig. B.10 and framework S-OUR, consisting of VAE₁₆- f - L_1 , LSR- L_1 and APN₁₆- f - L_1 , and compared the performance with the results from our earlier work [7] obtained using S- L_1 , S- L_2 and S- L_∞ .

Method	Syst.	Trans.	P_I	P_u
Fold 1 to 5 - comparison to [7]				
S-OUR	96%	99%	100%	100%
S- L_1 [7]	80%	90%	100%	100%
S- L_2 [7]	40%	77%	60%	60%
S- L_∞ [7]	24%	44%	56%	36%
Fold layer				
S-OUR	50%	83%	100%	100%

Table B.8: Results (best in bold) for executing visual action plans on 5 folding tasks (each repeated 5 times) shown in the top. The bottom row shows the results on the fold requiring to pick the top layer of the garment (repeated 10 times).

The results are shown in Table B.8, while, as previously mentioned, all execution videos, including the respective visual action plans, are available on the website¹. We report the total system success rate with re-planning, the percentage of correct single transitions, and the percentage of successful visual plans and action plans from start to goal. We observed that S-OUR outperformed the systems from [7] with a notable 96% system performance, only missing a single folding step which results in a transition performance of 99%. As for S- L_1 , S-OUR also achieved optimal performance when scoring the initial visual plans P_I and the initial action plans P_u . *We thus conclude that the improved MM, LSR and APM modules together contribute to a significant better system than in [7].*

10.5.2 Folding with multiple layers

As the previous folds resulted in nearly perfect performance of our system, we challenged it with an additional much harder fold that requires to pick the top layer of the garment. The fold, shown in Fig. B.10 bottom right, was repeated 10 times. An example of the obtained visual action plan is shown in Fig. B.13 and the final results are reported in Table B.8 (bottom row).

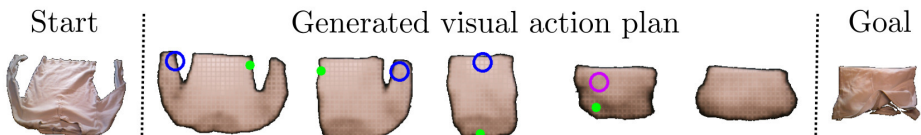


Figure B.13: Visual action plan for the fold requiring to pick the top layer of the garment. The step where the top layer is to be picked is indicated in purple (see accompanying video for further details).

Experiments showed that the system had no trouble *planning* the folding steps

from the initial configuration and was able to properly plan layer folds (with pick location marked in purple). Concerning the *execution* of the plan, the robot managed to correctly fold in 80% of the cases, excluding the last fold, using the re-planning strategy. However, failure cases often occurred during the execution of the last layer fold, resulting in the robot picking up multiple layers at the same time. When this happened, the T-shirt deformed into unseen states that were very dissimilar from the ones in \mathcal{T}_I and that rendered the re-planning step inefficient. A more precise manipulation system, either using a specialized gripper or custom methods for separating cloth layers, could potentially boost the performance of our system on this specific folding task. We leave these improvements for future work.

11 Conclusions

In this work, we presented an extended version of the Latent Space Roadmap first introduced in [7] which allows visual action planning of manipulation tasks. Firstly, we improved the building procedure of the LSR in the latent space by introducing an outer optimization loop that eliminates the need for a hard-to-tune clustering parameter. Secondly, we improved the training procedure of the VAE, used to represent the Mapping Module, by dynamically increasing the desired distance between action pairs. We thoroughly investigated the structure of the latent space, and presented a deep insight into the effects that each of the improvements have for the system. In addition, we compared different realizations of the Action Proposal Module and showcased the benefits of using latent representations for generating action plans. Lastly, we evaluated the LSR on three simulated tasks as well as real-world folding task. We introduced a harder version of the box stacking task and a rope-box manipulation task involving a rigid and deformable object, which enabled a more informative ablation study. We showed that the improved LSR significantly outperforms the one presented in [7] on the same folding task.

We are convinced that in order to advance state-of-the-art manipulation techniques for rigid and deformable objects, improvements on two fronts are necessary: learning a structured latent space as well as its exploration. We believe that our proposed method is a step toward achieving this goal which also opens many interesting future directions. For example, we wish to expand our method to encode full trajectories that could be leveraged to further structure the latent space, or to apply it to reinforcement learning settings with active exploration.

12 Appendix

This section provides complementary results on the separability of data samples for the simulation and the folding tasks, an ablation study considering different latent dimensions as well as a short overview of the different clustering methods.

ld	ns		hs		rb	
	VAE- b	VAE- L_1	VAE- b	VAE- L_1	VAE- b	VAE- L_1
4	73.9	159.7 ± 23.9	29.2	93.8 ± 10.1	23.4	97.6 ± 4.6
6	19.4	59.8 ± 4.1	11.7	39.2 ± 2.3	9.7	41.0 ± 5.1
8	11.7	45.6 ± 1.6	8.3	24.1 ± 1.8	6.3	30.7 ± 4.4
12	5.7	46.0 ± 7.0	4.8	20.6 ± 1.6	4.4	19.0 ± 1.9
16	4.4	40.4 ± 8.1	3.9	21.6 ± 2.3	3.8	18.8 ± 1.9
32	3.7	46.0 ± 4.3	3.7	29.9 ± 3.9	3.1	17.0 ± 1.4

Table B.9: Evaluation of the relative contrast rc on \mathcal{T}_z obtained with baseline and action VAEs. Results for all simulated tasks are reported.

12.1 Separability of data samples

To further analyze how the action loss influences the structure of the latent space, we resorted to the concept of *relative contrast*, introduced in [47] and based on [48]. Given a dataset \mathcal{T} containing ld -dimensional data points, the relative contrast rc of \mathcal{T} with respect to the metric L_p is defined as

$$rc(\mathcal{T}, ld, p) = \frac{D_{\max} - D_{\min}}{D_{\min}} \quad (\text{B.11})$$

where D_{\max} and D_{\min} denote the distances of the furthest and the closest points in the dataset \mathcal{T} from a chosen point [47], respectively. We calculated rc of the latent training dataset \mathcal{T}_z using VAEs with different latent dimensions ld . The rc values were used to evaluate significance of distances among points in the latent space \mathcal{Z} , where a lower rc implies diminishing contrast in distances. Authors of [48] show that, under broad assumptions, rc converges to 0 when ld approaches infinity, as in this case the distance from a point to its nearest neighbor is identical to the one to its furthest neighbor.

In our case, measuring rc on the full dataset can be misleading since the latent no-action pairs are encoded closeby because of the action term (B.4). This would lead to a deceptively small D_{\min} and therefore to a higher rc than warranted. To get a reliable estimate of rc given \mathcal{T}_z , we therefore chose to measure it in the following way. We split the latent training dataset \mathcal{T}_z into datasets \mathcal{T}_z^1 and \mathcal{T}_z^2 containing the first and the consecutive states of all the training tuples in \mathcal{T}_z , respectively. We then measured D_{\min} and D_{\max} with respect to each state in \mathcal{T}_z^1 and \mathcal{T}_z^2 separately. More precisely, we considered each latent state $z_i \in \mathcal{T}_z^1$ as the origin, and calculated D_{\min} (and D_{\max}) within $\mathcal{T}_z^1 \setminus \{z_i\}$. We repeated the same process for all $z_i \in \mathcal{T}_z^2$. Finally, we used the average of the obtained D_{\min} and D_{\max} values in the computation of rc as in (B.11).

Simulation tasks: Table B.9 reports the relative contrast of \mathcal{T}_z for all simulation tasks. We observed that VAE $_{ld-L_1}$ achieved a higher rc than the baseline VAE $_{ld-b}$. This is because the action loss (B.4) encourages no-action pairs to be encoded closeby, hence minimizing D_{\min} , and action pairs to be encoded at least d_m apart,

hence encouraging D_{\max} to be at least d_m . From the same reason, rc decreased with increasing ld for baseline VAEs, while it was rather constant for the action models VAE_{ld-L_1} . Moreover, we obtained higher rc values for ns than hs or rb which once more highlights the differences in difficulty between the three tasks.

Folding task: Table B.10 shows the relative contrast rc of \mathcal{T}_z produced by the baseline models VAE_{16-f-b} (left) and action models VAE_{16-f-L_1} (right) when varying the latent space dimension ld . For the action models (right) we report the mean and standard deviation of rc computed over 5 seeds.

ld	VAE_{ld-f-b}	VAE_{ld-f-L_1}
4	24.2	37.7 ± 4.3
6	13.7	26.1 ± 2.7
8	10.2	22.3 ± 2.2
12	7.2	15.8 ± 3.2
16	5.8	12.1 ± 1.2
32	4.5	8.2 ± 0.8
64	3.7	8.1 ± 0.8

Table B.10: Relative contrast evaluation comparing baseline (left) and action VAEs (right) when varying ld .

We observed that the action term (B.4) significantly increased the relative contrast for all tasks but clearly dropped for higher latent dimensions. We thus conclude that the action term and the dynamic setting of d_m improve the relative contrast.

12.2 Latent space dimension

The problem of choosing a suitable latent space dimension has not received much attention in the literature. Even though the action term alleviates the problem of indistinguishable distances in higher dimensions by increasing the relative contrast, it is still important to choose a latent dimension large enough so that the relevant features can be encoded. In Table B.11 we report the partial scoring on normal and hard stacking and rope-box tasks using VAE models with various latent dimensions. The results demonstrate an evident drop in the performance when the latent dimension was too small, such as $ld = 4$. As ld increased, we observed gradual improvements in the performance where a satisfactory level was achieved using $ld \geq 6$ for ns , and $ld \geq 12$ for hs and rb . Therefore, hs and rb required more dimensions in order to capture all the relevant and necessary features. *This result not only demonstrates the complexity of each task version but also justifies the choice $ld = 12$ in the simulations.*

ld	ns [%]	hs [%]	rb [%]
4	7.9 ± 2.2	8.8 ± 7.9	62.7 ± 13.9
6	99.96 ± 0.08	56.2 ± 23.1	74.9 ± 5.0
8	99.96 ± 0.08	62.7 ± 18.7	80.6 ± 5.3
12	100.0 ± 0.0	92.1 ± 2.9	90.4 ± 2.9
16	100.0 ± 0.0	95.9 ± 1.4	92.2 ± 1.1
32	97.5 ± 4.33	96.4 ± 0.4	92.6 ± 2.0

Table B.11: Comparison of the LSR performance when using VAEs with different latent dimensions for all the simulated tasks.

12.3 Overview of clustering algorithms

In this section, we provide a brief overview of the ablated clustering methods considered in Sec. 9.3.4.

Epsilon clustering: used in our earlier work [7] and functionally coincident with DBSCAN [57]. Its performance is affected by the parameter ε , i.e., radius of the ε -neighborhood of every point, and deteriorates when clusters have different densities.

Mean-shift: centroid-based algorithm [51] with moving window approach to identify high density regions. At each iteration, the centroid candidates associated to the windows are updated to the mean of the points in the considered region. The window size has a significant influence on the performance.

OPTICS: improved version of DBSCAN introduced by [52] in which a hierarchical reachability-plot dendrogram is built, whose slope identifies clusters with different densities. The parameter $\Xi \in [0, 1]$ is used to tune the slope and heavily affects the outcome of the algorithm. However, its influence is not easy to understand intuitively, as discussed in [58].

Linkage: hierarchical, agglomerative clustering algorithm discussed in Sec. 6.1. Possible dissimilarity functions to merge points are *single*, based on the minimum distance between any pair of points belonging to two distinct clusters, *complete*, based on the maximum distance, and *average*, based on the unweighted average of the distances of all points belonging to two distinct clusters. As discussed in Sec. 6.1, the clustering threshold τ determines the vertical cut through the dendrogram and consequently influences the performance of the algorithm.

HDBSCAN: agglomerative clustering algorithm in which the branches of the dendrogram are optimized for non-overlapping clusters using a notion of "cluster stability" based on their longevity. HDBSCAN automatically identifies clusters with different densities and requires specifying only the minimum cluster size prior to the training.

References

1. J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, pp. 2863–2871, 2015.
2. I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Aleny , and D. Kragic, "Benchmarking bimanual cloth manipulation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1111–1118, 2020.
3. H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robot.*, vol. 6, no. 54, 2021.
4. D. H. Ballard, "Modular learning in neural networks.," in *AAAI*, pp. 279–284, 1987.
5. D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *Int. Conf. Learn. Represent.*, 2015.
6. D. M llner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv:1109.2378*, 2011.
7. 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," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 5619–5626, 2020.
8. N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *Int. Conf. Learn. Represent.*, 2018.
9. 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. Mach. Learn.*, pp. 2555–2565, 2019.
10. T. Lozano-P rez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 3684–3691, IEEE, 2014.
11. C. Galindo, J.-A. Fern ndez-Madrigal, J. Gonz lez, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955 – 966, 2008. Semantic Knowledge in Robotics.
12. L. P. Kaelbling and T. Lozano-P rez, "Integrated task and motion planning in belief space," *Int. J. Robot. Res.*, vol. 32, no. 9-10, pp. 1194–1227, 2013.
13. S. M. LaValle, *Planning Algorithms*. Cambridge U.K.: Cambridge University Press, 2006.
14. R. Bellman, "Curse of dimensionality," *Adaptive control processes: a guided tour. Princeton, NJ*, vol. 3, p. 2, 1961.
15. A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks," in *Int. Conf. Mach. Learn.*, 2018.

16. B. Jia, Z. Pan, Z. Hu, J. Pan, and D. Manocha, "Cloth manipulation using random-forest-based imitation learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2086–2093, 2019.
17. A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Trans. Robot.*, pp. 1–19, 2020.
18. J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conf. Robot Learn.*, pp. 734–743, PMLR, 2018.
19. R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation," in *Robotics: Science and Systems (RSS)*, 2020.
20. X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conf. Robot Learn.*, 2020.
21. R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner, "Learning arbitrary-goal fabric folding with one hour of real robot experience," *Conf. Robot Learn.*, 2020.
22. B. Ichter and M. Pavone, "Robot Motion Planning in Learned Latent Spaces," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2407–2414, 2019.
23. M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.
24. C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conf. Robot Learn.*, pp. 1113–1132, PMLR, 2020.
25. A. Wang, T. Kurutach, P. Abbeel, and A. Tamar, "Learning robotic manipulation through visual planning and acting," in *Robotics: Science and Systems*, 2019.
26. T. Kurutach, A. Tamar, G. Yang, S. J. Russell, and P. Abbeel, "Learning plannable representations with causal infogan," in *Advances in Neural Information Processing Systems*, pp. 8733–8744, 2018.
27. A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *IEEE Int. Conf. Robot. Autom.*, pp. 2146–2153, 2017.
28. W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," *Conf. Robot Learn.*, 2020.
29. K. Pertsch, O. Rybkin, F. Ebert, C. Finn, D. Jayaraman, and S. Levine, "Long-horizon visual planning with goal-conditioned hierarchical predictors," in *Advances in Neural Information Processing Systems*, 2020.
30. C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *IEEE Int. Conf. Robot. Autom.*, pp. 2786–2793, 2017.

31. T. Kipf, E. van der Pol, and M. Welling, “Contrastive learning of structured world models,” in *Int. Conf. Learn. Represent.*, 2020.
32. B. Eysenbach, R. R. Salakhutdinov, and S. Levine, “Search on the replay buffer: Bridging planning and reinforcement learning,” in *Advances in Neural Information Processing Systems*, pp. 15246–15257, 2019.
33. R. Li, A. Jabri, T. Darrell, and P. Agrawal, “Towards practical multi-object manipulation using relational reinforcement learning,” *IEEE Int. Conf. Robot. and Automation*, pp. 4051–4058, 2020.
34. R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 1735–1742, 2006.
35. D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *Int. Conf. Mach. Learn.*, pp. 1278–1286, 2014.
36. I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “ β -vae: Learning basic visual concepts with a constrained variational framework,” *Int. Conf. Learn. Represent.*, 2017.
37. C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, “Understanding disentangling in β -vae,” *arXiv preprint arXiv:1804.03599*, 2018.
38. R. R. Sokal, “A statistical method for evaluating systematic relationships,” *Univ. Kansas, Sci. Bull.*, vol. 38, pp. 1409–1438, 1958.
39. M. E. Celebi, *Partitional clustering algorithms*. Springer, 2014.
40. P. Langfelder, B. Zhang, and S. Horvath, “Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r,” *Bioinformatics*, vol. 24, no. 5, pp. 719–720, 2008.
41. D. Bruzese and D. Vistocco, “Despota: Dendrogram slicing through a permutation test approach,” *J. Classif.*, vol. 32, no. 2, pp. 285–304, 2015.
42. A. Pasini, E. Baralis, P. Garza, D. Floriello, M. Idiomi, A. Ortenzi, and S. Ricci, “Adaptive hierarchical clustering for petrographic image analysis,” in *EDBT/ICDT Workshops*, 2019.
43. R. P. Brent, “An algorithm with guaranteed convergence for finding a zero of a function,” *Computer J.*, vol. 14, no. 4, pp. 422–425, 1971.
44. G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, “Active incremental learning of robot movement primitives,” in *Conf. Robot Learn.*, pp. 37–46, 2017.
45. Unity Technologies, “Unity.”
46. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

47. C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Int. Conf. Database Theory*, pp. 420–434, Springer, 2001.
48. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?," in *Int. Conf. Database Theory*, pp. 217–235, Springer, 1999.
49. Z. Xie, I. Sato, and M. Sugiyama, "Stable weight decay regularization," *arXiv preprint arXiv:2011.11152*, 2020.
50. R. Berk, "A primer on robust regression," *Modern methods of data analysis*, pp. 292–324, 1990.
51. Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, 1995.
52. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.
53. W. H. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *J. Classif.*, vol. 1, no. 1, pp. 7–24, 1984.
54. L. McInnes, J. Healy, and S. Astels, "HDBSCAN: Hierarchical density based clustering," *J. of Open Source Software*, vol. 2, no. 11, p. 205, 2017.
55. C. Burgess and H. Kim, "3d shapes dataset." <https://github.com/deepmind/3dshapes-dataset/>, 2018.
56. J. Qian, T. Weng, L. Zhang, B. Okorn, and D. Held, "Cloth region segmentation for robust grasp selection," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 9553–9560, IEEE, 2020.
57. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.
58. R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pp. 160–172, Springer, 2013.

Paper C

Paper C

Benchmarking Bimanual Cloth Manipulation

Irene Garcia-Camacho*, Martina Lippi*, Michael C. Welle, Hang Yin, Rika Antonova, Anastasia Varava, Julia Borrás, Carme Torras, Alessandro Marino, Guillem Alenyà and Danica Kragic

Abstract

Cloth manipulation is a challenging task that, despite its importance, has received relatively little attention compared to rigid object manipulation. In this paper, we provide three benchmarks for evaluation and comparison of different approaches towards three basic tasks in cloth manipulation: spreading a tablecloth over a table, folding a towel, and dressing. The tasks can be executed on any bimanual robotic platform and the objects involved in the tasks are standardized and easy to acquire. We provide several complexity levels for each task, and describe the quality measures to evaluate task execution. Furthermore, we provide baseline solutions for all the tasks and evaluate them according to the proposed metrics.

1 Introduction

Manipulation of highly deformable objects, such as cloth, is an important area of robotics research that has applications both in industrial scenarios and in domestic environments. Despite its relevance, this research direction has historically received relatively little attention compared to rigid object manipulation due to the challenges it entails. Recently, a stronger interest in deformable object manipulation emerged and the survey in [1] presents the latest advances.

* Authors contributed equally, listed in alphabetical order.

In order to effectively evaluate robotics methods, it is beneficial to provide specialized benchmarks [2]. A benchmark is a set of well-defined tasks to be performed in a standardized setup which needs to be easy to reproduce in different robotics laboratories. Existing manipulation benchmarks include large object sets [3], unified protocol procedures [4], robotic competitions [5], task specific benchmarks such as [6] for the picking task and also manipulation task taxonomies [7]. However, to the best of our knowledge, they all involve only rigid objects. In this paper, we provide benchmarks that will help assess the capability of a robotic system for manipulation of cloth-like objects.

To handle high degree of uncertainty about deformable objects’ state, perception and manipulation often need to be intertwined. Furthermore, the choice of grasping and re-grasping strategies can significantly impact subsequent manipulation. Thus, one challenge in designing a benchmark for cloth manipulation is that different components of a robotic system, such as perception, grasping and manipulation planning, are highly dependent on each other. Therefore, we propose to evaluate the performance of the entire system rather than evaluating perception versus action components separately. We also recognize that for some tasks it is common to treat grasping of the initial target points as a sub-task. Hence, we provide a way to evaluate grasp execution, followed by evaluation of the task after the cloth is grasped.

We propose three benchmarks corresponding to three evaluation tasks that, in our opinion, form a basis for more complex tasks for handling clothes and dressing a human. *(i)* The first task is unfolding a tablecloth and spreading it over a table. The need to spread cloth-like objects, such as bed sheets and tablecloth, is ubiquitous in our everyday life. In addition, this task can be seen as preparatory, e.g spreading on a flat surface for ironing or folding. *(ii)* The second task is folding a towel on a table. This is one of the most common tasks in textile manipulation literature [8], and can be seen as a preparatory action before placing on a shelf or in a box for storage/packaging. Although prior works proposed methods for folding [9, 10, 11, 12, 13], these have never been systematically compared or benchmarked. *(iii)* The third task consists of fitting the neck of a T-shirt over a 3D printed head. This task is a simplification of a dressing scenario: a basis for more complex tasks like putting a T-shirt or a sweater on a human or mannequin.

We define performance metrics to evaluate a cloth manipulation method which are based on success of the task, execution time, force measures and, if possible, quality of the final result, e.g., we define how a tablecloth should be placed on the table. In this way, each proposed benchmark is well-equipped to distinguish approaches that are likely to perform general cloth manipulation well. Finally, we provide baseline solutions for all the tasks and evaluate them according to the proposed scoring, recording decreasing success rate as the complexity of scenarios increases.

Despite initial progress, cloth manipulation remains mostly unsolved, with innovative techniques still under development. Our proposed benchmarks form a systematic testbed for the prototypical cloth manipulation tasks, helping to evalu-

Code	Objects for manipulation	
	Tablecloth	IKEA Fullkomlig 1.45×2.4 m
[st]	Small towel	IKEA towel Hären 0.3×0.5 m or 0.3×0.3 m
[bt]	Big towel	IKEA towel Hären 0.5×1 m or 0.4×0.7 m
	T-Shirt	Any T-shirt in accordance to Figure C.1
Code	Environmental objects	
	Table	Any table with dimension in the range Length: $[1.2, 1.85]$ m Width: $[0.7, 0.8]$ m Height: $[0.72, 0.75]$ m
[sh]	Small head	Generate 3D model with provided script
[bh]	Big head	Generate 3D model with provided script

Table C.1: List of objects with instructions for acquisition

ate emerging approaches and to gain technical insights for further improvement.

2 The Benchmarks

We propose three benchmarks for manipulation of highly deformable cloth-like objects which can be performed by any bimanual setup. In particular, our benchmarks focus on three basic tasks in cloth manipulation which involve textile objects of different sizes and types: spreading a tablecloth, folding a towel, and dressing. To foster easy modular use of the benchmarks, we separate each task into sub-tasks that can be evaluated individually, varying the level of difficulty on the basis of the cloth initial configuration and involved objects. Protocols (RAL-SI-2020-P19-0832_1-V1.0, RAL-SI-2020-P19-0832_2-V1.0, RAL-SI-2020-P19-0832_3-V1.0 for the three tasks, respectively) can be found in the attached material and at the link¹ with their respective explanatory videos and benchmark documents.

In the following we give a summary of the benchmarks tasks, setup and evaluation. For the sake of clearness, setup, sub-task decomposition and evaluation are common to three benchmarks and will thus be presented jointly. Further information can also be found on the website¹.

2.1 Tasks description

2.1.1 Task 1: Spreading a tablecloth

This task consists of grasping a tablecloth and spreading it on a table, using the table and the tablecloth indicated in Table C.1. An example of implementation is shown in Figure C.2-left.

Similarly to the other tasks, this task requires to grasp the cloth at two grasping points, usually two of the corners, and then to manipulate it to spread it. For the first grasp, we consider different starting cloth configurations: from folded to

¹<https://ral-si.github.io/cloth-benchmark/>

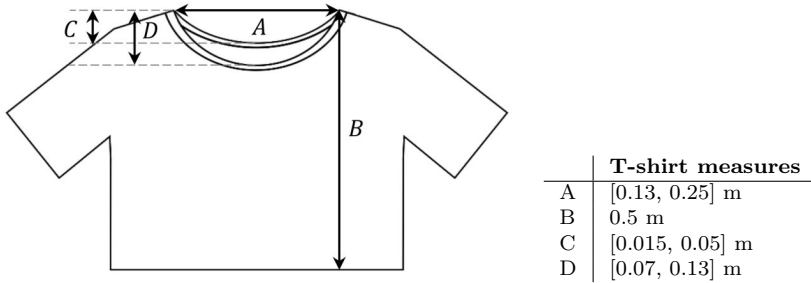


Figure C.1: Representation of the allowed measures for the T-shirt; B measure is fixed to equalize the level of difficulty when performing the dressing.



Figure C.2: From left to right, example of implementations of tasks one to three, respectively.

crumpled on the table (see Figure C.3). Grasping crumpled cloth at a desired grasping point has been attempted many times in literature by localizing corners or edges [9, 14] or more specific parts [15, 16, 17]. In contrast, starting from a folded configuration was rarely considered, despite it being a common cloth state in domestic environments. The challenge in this case lies in grasping just one single corner of the many layers that are folded together. After the first grasp, the cloth needs to be grasped at the second grasping point to unfold the cloth. Then it needs to be spread on the table. The task requires manipulating a big piece of cloth, which is challenging for many robots and may call for additional strategies. Overall, our protocol does not impose a specific strategy. This gives more freedom to researchers to develop and compare innovative approaches.

2.1.2 Task 2: Folding a towel

This task consists of grasping a towel and folding it. The task uses the same table as the previous task and two different sizes of towels, as indicated in Table C.1. An example of implementation is shown in Figure C.2-middle. This is a classic cloth manipulation task. Since the early example of PR2 robot folding towels in 2010 [9], there have been many other works focusing on folding towels and other items [11, 18, 12, 19, 20]. However, as stated in Section 1, this task has never been

benchmarked or properly compared based on the quality of the folds or execution time.

Folds location varies significantly depending on the garment geometry [13]. Even for a rectangular napkin or towel, there are multiple fold strategies one could follow. However, once the fold line has been decided, we want to focus on finding manipulation that can best realize it. For this reason, we focus on the simplest strategy for rectangular items: always fold in half and perform a maximum of three folds which we evaluate individually. This strategy has the advantage that it can be easily evaluated by taking top view snapshots after every fold. Besides the starting crumpled configuration (as in Task 1), we also consider a ‘flat on the table’ configuration, also shown in Figure C.3.

When the cloth is crumpled, the main difference for grasping, compared to Task 1, is the size of the object. A small/medium versus large size would entail the need for different strategies to enable initial grasping. We do not impose a specific folding strategy. Small towels can be folded using the classic strategy of placing them on a table and picking two corners to fold [9, 12], but bigger ones might require alternatives [21].

2.1.3 Task 3: Partial dressing

The goal of the third task is to put a T-shirt over a simple head model starting from different initial configurations of the garment, as shown in Figure C.2-right. Putting on the sleeves is not included in the task. The complex geometric shape of T-shirts makes their manipulation towards desired states a difficult process that requires a tight integration of perceptive sensors, such as cameras and force/torque sensors, into the manipulation strategy. Thus, the focus of this task lies in evaluating the combination of perception and manipulation strategies.

Analogously to the previous tasks, the success of the manipulation task highly depends on the way the garment is grasped, therefore we consider several initial configurations of the T-shirt which allow to explore different grasping strategies: crumpled, flat or folded on the table. Previous approaches to this task used reinforcement learning and topology coordinates [22, 23], but with a different starting configuration and assuming the cloth is pre-grasped.

Another important aspect is the relative size of the head with respect to the collar circumference: the larger the head is, the harder it is to execute the task. For this reason, we provide head models of two different sizes as reported in Table C.1. Finally, to avoid damaging the head and the garment, it is desirable to monitor applied forces especially when the collar is tight.

2.2 Setup description

2.2.1 Hardware description

Any bimanual setup with grasping capabilities can be employed and any sensor that can aid in completing the task is allowed (e.g., RGBD-cameras).



Figure C.3: Examples of starting configurations with a towel: **[fd]**: folded on the left, **[cr]**: crumpled on the middle and **[ft]**: flat on the right.

2.2.2 Objects description

Table C.1 lists all the objects involved in the tasks with the link or information to acquire them. The YCB object set [4] includes two cloth items, that are a tablecloth and a T-shirt. The YCB tablecloth is designed to cover a standard 1.8 m long table until the floor. However, we propose a smaller IKEA tablecloth because its size is already challenging for the current state of the art. Regarding the YCB T-shirt, we observed how even T-shirts from the same batch have a high variance of measures. This also holds true for other available T-shirts. Therefore, we define a range of measures that are accepted for the T-shirt as reported in Figure C.1. In this way, greater flexibility is guaranteed compared to the case of a predefined single T-shirt and the possibility of adopting the benchmark is maintained despite continuous changes in fashion. Any size from S to XL of the YCB T-shirt should fall into the allowed range. The length of the T-shirt (measure B in Figure C.1) is fixed to allow comparability of different methods, since it determines the amount of garment that needs to pass through the head. Note that every T-shirt that is used needs to be measured even if it comes from the same batch to account for production variance. Finally, concerning the towel for Task 2, we include two sizes, a small towel (**[st]**) and a bigger one (**[bt]**). However, the sizes of the big and small towels slightly change depending on the country, therefore, we provide two options for the small towel and two for the big towel. Note that the use of big towels is already a step forward in the literature in terms of object size.

In addition, two environmental objects are required, a table (for Tasks 1 and 2) and a human-like head (for Task 3). Following the idea of flexibility to make the setup easy to reproduce, we do not fix lightning conditions and we do not provide a specific table model but just an interval of table sizes. Concerning the human-like head, two different sizes are considered which are small (**[sh]**) and big (**[bh]**) and their models are defined according to the T-shirt measures. A script is provided in the attached material to automatically generate the 3D model. Refer to protocols for further details.

2.2.3 Initial cloth configuration descriptions

In general, when a task on cloth manipulation is attempted, the initial state of the cloth falls in one of these categories:

[pg2] Cloth is pre-grasped at two points.

[pg1] Cloth is pre-grasped at one point.

[ft] Cloth is lying flat on a table (Figure C.3-right).

[fd] Cloth is folded on a table (Figure C.3-left).

[cr] Cloth is crumpled on a table (Figure C.3-middle).

These starting configurations will be common for all the protocols benchmarking each task, although not all starting configurations are used for all task. For instance, it is pointless to consider the folded configuration for the folding task.

We will refer to the parts of the cloth that need to be grasped as grasping points. In a towel, the grasping points are usually the corners but they can be redefined; instead, for a T-shirt, these strictly depend on the manipulation strategy.

2.3 Sub-Tasks description

Given a task, the respective sub-tasks are obtained by considering all the possible combinations of involved objects and initial cloth configurations: a tablecloth with 4 initial configurations for Task 1, two towels with 4 initial configurations for Task 2 and two head sizes with 5 initial configurations for Task 3.

In addition, each sub-task can be decomposed in the following phases:

[GR1] Grasp first grasping point.

[GR2] Grasp second grasping point with other hand.

[MAN] Perform the manipulation (depending on the task).

Note that both [GR1] and [GR2] may require manipulation; for instance, in order to grasp a crumpled cloth from a table and reach the first grasping point, the cloth may need to be pre-manipulated, and all this actions constitute the [GR1] phase. Obviously, no [GR1] and [GR2] phases are required in case of starting configuration [pg2] as well as no [GR1] phase is executed for the initial configuration [pg1].

Users can submit all phases of one sub-task, or just one phase alone. This sub-division in phases and sub-tasks allows to achieve incremental complexity, letting the user choose the desired level of difficulty to face, e.g., dressing task with small head and [pg2] initial configuration is clearly less challenging than the case of big head with [fd] initial configuration.

2.4 Evaluation of results

To enhance progress, allow reproducibility of results and easy comparison between different works, we propose the following list of performance metrics: *success of each phase*, *execution time*, *force measures* (if available) and *quality measures*. The choice not to provide a single value to assess goodness but a set of values is motivated by the fact that, in such complex tasks, the former may be too reductive; in this way, instead, each user can focus on the aspects of interest, e.g solutions that require longer time but exert lower forces. In the following, the proposed performance metrics are detailed.

2.4.1 Success of each phase

In light of the phases subdivision in Section 2.3, each phase can be evaluated individually in regards to completeness. Phases [GR1] and [GR2] are considered successfully completed if the grasping is performed and is held during the whole manipulation and, in Tasks 1 and 3, if the cloth is unfolded with starting configuration [fd]. The condition of success for phase [MAN] depends on the considered task: the tablecloth is successfully spread if it covers the table top; the folded towel is successfully folded if one fold is done and opposing corners are together (each fold is evaluated individually); for the dressing task we assume this phase is accomplished when the neck hole of the T-shirt is put over the head and the entirety of the T-shirt lies below the head. We let the users define different grasping points according to their strategy, not to limit the possible approaches in the [MAN] phase. In case the manipulation phase is successful, the grasping phases will be considered successful in turn.

To increase the flexibility of the benchmarks and promote participation, we leave the freedom to report only the manipulation part [MAN], which may be the case for end-to-end learning-based approaches, or only the grasping part [GR1] and [GR2] if a group is strong in grasping but lacks the perception solutions to successfully execute the manipulation. We believe this can be valuable to share solutions and combine different approaches to push the solutions forward.

2.4.2 Execution time

The execution time comprises the times needed for all the phases, and it is measured from the moment the first robot starts to move until the end of the manipulation.

2.4.3 Force measures

Force measures at the end effectors quantify the interaction between the robots and the environment; they are only acquired during phase [MAN] and minimum, maximum and average norms are considered. Note that, in order not to limit the possibility of using the benchmarks, force measures are not mandatory but are

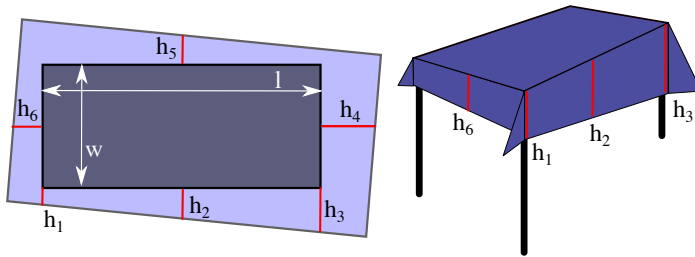


Figure C.4: Representation of the measures to evaluate how well the tablecloth has been placed.

highly encouraged especially in dressing task, where monitoring of exerted forces on the head represents a key feature.

2.4.4 Quality measures

For the tasks of tablecloth spreading and folding, the quality of the result of the execution can be measured, e.g., poor results are achieved if the tablecloth is completely tilted or if towels are folded wrinkly or with the corners not matching.

To take that into account, we define a quality function that measures the percentage of error of the task result. Note that for the dressing task, no measures can be defined because of the binary nature of the task.

Quality measures for Task 1

We evaluate how much the tablecloth is rotated and translated with respect to the table. To this aim, as represented in Figure C.4, a total of 6 tablecloth drop lengths at different sides of the table need to be measured after the tablecloth is spread. Measures can be taken from the middle of each table edge. For a table with length t_l and width t_w and a tablecloth with length c_l and width c_w , the proposed percentage of errors are:

$$\begin{aligned} \text{\% rotation error: } E_\alpha &= \frac{\arctan\left(\frac{|h_3-h_1|}{t_l}\right)}{\pi/4} \\ \text{\% length translation error: } E_l &= \frac{|h_6-h_4|}{c_l-t_l} \\ \text{\% width translation error: } E_w &= \frac{|h_2-h_5|}{c_w-t_w} \end{aligned} \quad (\text{C.1})$$

These quality functions can only be applied if the task has been successfully accomplished, meaning that the tablecloth is covering all the table top. Then, a 100% rotation error occurs when the tablecloth is rotated by $\pi/4$ radians (45°), which is unlikely to happen if the tablecloth is fully covering the table. The maximum translation error occurs when one of the hanging parts is zero, meaning the table is

almost uncovered. If the hanging part of the table cloth is touching the floor, one needs to measure the tablecloth drop length ignoring the floor. If any circumstance occurs (e.g., one of the hanging parts of the tablecloth is wrinkled), one should report this with a picture, even if it does not affect the quality function. Note that this error measure is independent of the size of the table and tablecloth, thus allowing a fair comparison among different setups.

Quality measures for Task 2

We assume a one fold manipulation is successful if the corners of the original spread cloth are matching two by two. That means if one of the corners is bent, we assume the robot should correct that, otherwise the task cannot be reported as a success. In addition, we measure how well the corners match by evaluating the ratio between the surface of the spread cloth before and after the fold. Because our task is restricted to folding in half, each fold needs to cut in half the area of the spread cloth on the table. This has to be measured at each fold either by measuring manually the area or by automatically computing it with a top view image.

Then, the proposed quality function for this task is

$$\% \text{ of error in a fold } E_f = \frac{100}{0.5} \cdot \left\| \frac{A_f}{A_i} - 0.5 \right\|, \quad (\text{C.2})$$

where A_f is the final area of the cloth from the top view, and A_i is the initial area of the cloth. Assuming A_f will always be smaller than A_i , 100% error occurs when $A_i = A_f$, but also if A_f is less than half of A_i , which can only happen if there are wrinkles or extra folds.

We promote the use of vision software to assess the area or the wrinkle state² of the towel [24].

2.5 Reporting results

Based on the above, we require that, for each sub-task, five trials are performed and then, for each trial, measures 2.4.1-2.4.4 are acquired. In addition, videos of the experiments and snapshots (or equivalent stylized figures) clearly representing the grasping points must be provided. A summary table, as shown in Table C.2, must be filled where, given a starting configuration, the success rate of each phase and average and variance of execution time, force measures and quality functions over the five trials are reported. When necessary, the size of the different elements must be reported as well, that are the table size for Tasks 1 and 2, the towel size for Task 2, and the head size for Task 3. Note that, in the folding task, results associated with each fold must be provided and top view pictures of each fold state have to be reported. Moreover, in order to assess the generality of the proposed approach, it is required to specify which assumptions (in a set in the respective scoring sheet) are made for completing the task, e.g., knowledge of the cloth color and pattern. In the case new assumptions are considered with respect to those in

²https://gitlab.iri.upc.edu/labrobotica/algorithms/finddd_descriptor

the scoring sheets, a detailed description on how they affect the solution must be reported. Finally, a discussion on:

- Employed hardware/software setup with specification of robots' details and respective number of motors;
- What makes the system successful;
- What makes the system fail;
- What is improved compared to other methods;

should be provided. A thorough description for the scoring of each task can be found in the provided Benchmark documents¹.

3 Baseline Systems

To showcase how the presented Benchmarks should be used and promote comparison of different methods, we describe our own systems tackling the Benchmarks.

3.1 Task 1: Spreading a tablecloth

The robotic system used for the baseline solution for Task 1 (and also 2) is composed of two TIAGo robots (shown in Figure C.2-left). Because the tablecloth size is large, we can take advantage of the base mobility. The arms are equipped with a modified parallel gripper that is flexible when it touches the table to allow to safely contact the table before grasping but rigid in the grasped direction. A table with measures $1.20 \times 0.7 \times 0.73$ m is used.

The solution for [GR1] to unfold the tablecloth is to grasp the first grasping point and pull the cloth up (Figure C.5-left). The solution for [GR2] is to grasp an edge point next to the first hand (Figure C.5-right) and trace the edge until the corner is reached. This implies sliding the cloth inside the gripper without losing it. This manipulation has been previously applied only to very small clothes in [10], and there are some cloth specialized grippers designed to ease this manipulation [25]. Each grasp is performed by a different TIAGo robot, and after they have the tablecloth grasped, they move across the table to spread the tablecloth. Note that this strategy is applicable to different sized tables because bimanual manipulation is achieved with two independent robots.

Our method depends on some simplifying assumptions, reported in the Benchmark sheet. We assume the folded piece is oriented on the table so that the grasping point is the closest to the robot. However, the tablecloth can be placed anywhere on the table. The second grasp in [GR2], shown in Figure C.5-right, is assumed to be at a fixed position with respect to the hand that is already grasping. This holds true for most of the cases, but may fail when the cloth is twisted differently than expected. Finally, the robot knows the size of the cloth and the table, so that we

Start. Config.	Mean quality func. (E_a, E_l, E_w)	Success			Time [s]
		[MAN]	[GR2]	[GR1]	
[pg2]	(1.44%, 8.23%, 17.67%)	80%	80%	18.28	
[pg1]	(1.61%, 7.50%, 46%)	60%	80%	72.24	
[cr]	-	0%	0%	-	
[fd]	-	0%	0%	-	

Towel size	Fold	First fold			Mean quality func. E_f	Time [s]
		[MAN]	[GR2]	[GR1]		
[pg2]	Success	80%	np	2.77%	24.35	
[pg1]	np	np	np	np	np	
[cr]	np	np	np	np	np	
[ft]	np	np	np	np	np	

Head Size	[sh]						[bh]							
	Success [MAN]	Success [GR2]	Success [GR1]	Time [s]	Force measures [N]			Success [MAN]	Success [GR2]	Success [GR1]	Time [s]	Force measures [N]		
Start. Config.					min	avg	max					min	avg	max
[pg2]	100%			29.30	0.54	3.02	7.29	100%			30.76	0.38	2.94	7.46
[pg1]	100%			65.48	0.49	3.89	9.34	80%			66.98	0.94	4.76	9.88
[cr]	40%	80%	80%	138.50	0.41	3.25	6.43	20%	60%	60%	127.4	0.39	4.32	8.15
[ft]	0%	60%	80%	-	-	-	-	0%	80%	100%	-	-	-	-
[fd]	0%	20%	60%	-	-	-	-	0%	20%	40%	-	-	-	-

Table C.2: Result summary tables. Notation “np” denotes that the respective sub-task has not been implemented in the baseline. For the sake of space, no variance values of execution times and forces are reported.

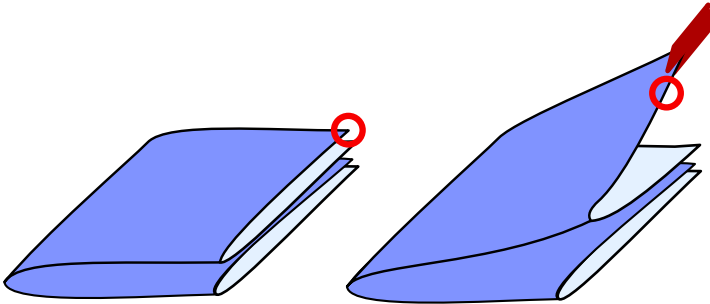


Figure C.5: The circles in red signal the location of the first grasping point for [GR1] (left) and on the initial grasp for the edge tracing in [GR2].

can estimate beforehand the amount of displacement needed when both the robot tracks the edge and puts the tablecloth.

3.2 Task 2: Folding a towel

For the task of folding a towel, we use the same robotic system as in the previous task and we consider the big towel ([bt]) with measures 0.5×1 m. The two mobile manipulators are placed at different sides of the table as shown in Figure C.2-middle. We only report the [MAN] phase, as the other phases are similar to those in Task 1. Thus, we start with the two corners already grasped, and we then perform the folding motion. For the folding strategy, we focus on the first fold and we use a Dynamic Movement Primitives (DMP) representation of the motion for each robot learned by demonstration, and execute both trajectories in synchronization. The size of the towel and the localization of the robot with respect to the table are assumed to be known.

3.3 Task 3: Partial dressing

For the dressing task, we propose a human inspired solution based on a vision/force-feedback informed strategy with hand-tuned hyperparameters. All the possible head sizes described in 2.2.2 and starting configurations in 2.2.3 are considered. Moreover, experiments with two T-shirts are carried out to show the validity of the protocol as long as the T-shirt complies with the range of measures provided. In detail, the following set of measures $\{A, B, C, D\}$ hold for the two T-shirts, respectively: $\{0.19, 0.5, 0.029, 0.1\}$ m and $\{0.154, 0.5, 0.025, 0.08\}$ m.

The robotic system, shown in Figure C.2-right, is composed of two Franka Emika Panda 7-DOFs manipulators equipped with parallel grippers. Customized long fingers have been adopted for one robot in order to have the fabric slipping into them during second grasping phase. The head is mounted on a podium stand in the middle of the workspace and its configuration is assumed to be known (in

particular, its position is represented by the upper point along the axis of symmetry). Moreover, a Logitech USB camera is mounted over the setup to provide a bird eye view of the workspace and link-side torque sensors at each link of the robots are available; based on these, an estimate of the forces exerted at the end effector of each robot is given. For details on sensors and estimates accuracy, the reader can refer to [26].

Concerning the grasping phases [GR1] and [GR2], pre-defined grasping poses are selected with all the initial configurations, thus no visual feedback is exploited in these stages. Concerning the manipulation phase [MAN], the formulation in [27] is leveraged for the dual-arm manipulation according to which the cooperative motion is expressed in terms of centroid and formation of the two end effectors. The basic idea of the devised strategy is to use the visual information to guide the team motion and, at the same time, to perform random wiggling motions which emulate human-like dressing. More specifically, the vision system splits the top-view circle associated with the head into two halves and measures the free area in each of them (see top right of Figure C.2-right). These measures are then exploited to determine in which direction to move the team centroid in such a way that both areas exceed a certain threshold. When the latter condition is fulfilled, the opening of the T-shirt is such that a sufficient surface of the head is visible through it, thus the downward motion to put on the T-shirt is started. In addition, small wiggling motions are introduced to facilitate the sliding of the garment along the head model. Finally, a continuous monitoring of end effector forces and of the elapsed time is performed: a restart procedure is planned when either force measurement exceeds a maximum allowed value or the elapsed time exceeds a time limit. For further details on the proposed baseline, the reader is referred to the document with solution comments in the accompanying material. Note that our baseline solution does not involve re-grasping phases but these are generally allowed.

4 Results

In this section, the evaluation of the baseline strategies according to the proposed Benchmarks is presented. Videos of each experiment and complete score sheets can be found in the results section at the website¹. A summary video is also provided in the accompanying multimedia material. For all the 3 tasks, performance results of the baseline solutions are shown in Table C.2.

4.1 Task 1: Spreading a tablecloth

Regarding the manipulation ([pg2] row), the mobile base of the robots is very effective. Only occasional entanglements of the tablecloth cause strong forces and make the grippers loose the garment.

The second grasp ([pg1] row) is quite robust because the strategy of following the edge has proven to be effective: the first interest point is always at the same point under the other robot gripper, and the edge tracing takes advantage of the

robot mobile torso to keep a vertical trajectory during as much time as possible. Failures are due to this last edge tracing phase: as the gripper has no force sensors, the edge is sometimes lost at the beginning of the movement.

Finally, in both **[fd]** and **[cr]** initial configurations, the first grasping phase resulted challenging. When **[fd]**, because sometimes several layers are grasped causing the second grasp to fail. The image used to locate the corner is taken from the head of the robot, and the viewpoint and distance make difficult to localize a single garment layer. When **[cr]**, because the friction of the fingers does not allow them to slide gently under the garment. In both cases, grasping fails and the task cannot be completed³.

4.2 Task 2: Folding a towel

Here we concentrate on the **[MAN]** itself and not on the **[GR1]** and **[GR2]** phases. Starting from initial configuration **[pg2]**, **[MAN]** achieves quite a good success ratio of 80% and we only detected one problem in the final release step. In particular, in one of the trials our grippers, that have high friction fingertips, and the towel remained stuck until the arm moved away, creating an unwanted bending. As corners are not matching two and two, and our system is not able to correct this, we evaluated this trial as failure³.

4.3 Task 3: Partial dressing

Performance results of the baseline solution with one T-shirt sample are summarized in Table C.2. In detail, success rates for the different phases are reported in case of both small and big heads which make evident the increasing complexity and challenges introduced by the different combinations of starting configurations and head sizes. Indeed, different starting configurations lead to a wide variety of achievable grasping points but our manipulation strategy is only able to deal with a subset of them; in particular, our manipulation strategy assumes that the T-shirt is grasped at two points along the neck collar and that one side of the T-shirt is enclosed in the fingers in such a way to minimize the amount of fabric hanging under them. Indeed, most manipulation failures with starting configurations **[ft]** and **[fd]** are due to the fact that there is too much fabric below the grasping points and the robots are unable to find an opening of the T-shirt to pass it through the head. In addition, blind pre-defined grasping poses make manipulation in **[cr]** configuration generally challenging. Finally, grasping failures are recorded with starting configurations **[fd]** when several layers of the T-shirt are grasped with the first gripper and unfolding does not happen. Table C.2 shows that the dressing task offers a wide range of possible improvements primarily in the manipulation phase but also in the grasping phases as each phase influences the following. For the sake of space, no further results are reported herein but complete scoring sheets and videos of the

³The complete score sheets can be found at <https://ral-si.github.io/cloth-benchmark/#results>

experiments are available at the link³, for the other T-shirt sample as well. Finally, it is worth remarking that, even in cases with a 100% success rate, there is still a considerable margin for improvement with respect to execution times and exerted forces reported in Table C.2.

5 Conclusions

In this paper, we proposed benchmarks for cloth manipulation with three representative tasks focusing on bimanual manipulation; they include cloth and garment items of various sizes. Each benchmark is hardware agnostic and flexible with respect to the strategies for solving the task. We believe that various robotics groups would find the benchmarks easy to use for comparing existing works and reporting new results. A simple well-defined object set and the possibility of reporting partial results make these benchmarks accessible to researchers targeting different stages of the tasks at various levels of difficulty. Our modular protocols also make the benchmarks potentially extendable to other tasks in the future.

We believe the baseline solutions give a valid initial point for comparison and show the increasing level of complexity of the different sub-tasks. Overall, this provides a good start for the research community to push the boundaries on what is possible in cloth manipulation further.

References

1. J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey,” *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 688–716, 2018.
2. F. Bonsignorio and A. P. del Pobil, “Toward replicable and measurable robotics research [from the guest editors],” *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 32–35, 2015.
3. B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The ycb object and model set: Towards common benchmarks for manipulation research,” in *Advanced Robotics (ICAR), 2015 Int. Conf. on*, pp. 510–517, 2015.
4. B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, “Benchmarking in manipulation research: Using the yale-CMU-berkeley object and model set,” *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 36–52, 2015.
5. F. Amigoni, E. Bastianelli, J. Berghofer, A. Bonarini, G. Fontana, N. Hochgeschwender, L. Iocchi, G. Kraetzschmar, P. Lima, M. Matteucci, P. Miraldo, D. Nardi, and V. Schiaffonati, “Competitions for benchmarking: Task and functionality scoring complete performance assessment,” *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 53–61, 2015.
6. J. Leitner, A. W. Tow, N. Sünderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, P. T. Kujala, and Lachlan, “The acrv picking

- benchmark: A robotic shelf picking benchmark to foster reproducible research,” in *IEEE Int. Conf. Robot. Autom.*, 2017.
7. A. H. Quispe, H. B. Amor, and H. I. Christensen, “A taxonomy of benchmark tasks for robot manipulation,” in *Springer Proceedings in Advanced Robotics*, pp. 405–421, Springer International Publishing, 2017.
 8. J. Borràs, G. Alenya, and C. Torras, “A grasping-centered analysis for cloth manipulation,” *arXiv:1906.08202*, 2019.
 9. J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, “Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding,” in *IEEE Int. Conf. Robot. Autom.*, pp. 2308–2315, 2010.
 10. H. Yuba, S. Arnold, and K. Yamazaki, “Unfolding of a rectangular cloth from unarranged starting shapes by a dual-armed robot with a mechanism for managing recognition error and uncertainty,” *Adv. Robot.*, vol. 31, no. 10, pp. 544–556, 2017.
 11. A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrik, A. Kargakos, L. Wagner, V. Hlavac, T.-K. Kim, and S. Malassiotis, “Folding clothes autonomously: A complete pipeline,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1461–1478, 2016.
 12. Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, “Folding deformable objects using predictive simulation and trajectory optimization,” in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 6000–6006, 2015.
 13. S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, “A geometric approach to robotic laundry folding,” *Int. J. Robot. Res.*, vol. 31, no. 2, 2012.
 14. Y. Li, D. Xu, Y. Yue, Y. Wang, S.-F. Chang, E. Grinspun, and P. K. Allen, “Regrasping and unfolding of garments using predictive thin shell modeling,” in *IEEE Int. Conf. Robot. Autom.*, pp. 1382–1388, 2015.
 15. C. Bersch, B. Pitzer, and S. Kammel, “Bimanual robotic cloth manipulation for laundry folding,” in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 1413–1419, 2011.
 16. A. Ramisa, G. Alenyà, F. Moreno-Noguer, and C. Torras, “Learning rgb-d descriptors of garment parts for informed robot grasping,” *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 246–258, 2014.
 17. E. Corona, G. Alenyà, T. Gabas, and C. Torras, “Active garment recognition and target grasping point detection using deep learning,” *Pattern Recognition*, vol. 74, pp. 629–641, 2018.
 18. M. Cusumano-Towner, A. Singh, S. Miller, J. F. O’Brien, and P. Abbeel, “Bringing clothing into desired configurations with limited perception,” in *IEEE Int. Conf. Robot. Autom.*, 2011.
 19. A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, “Learning force-based manipulation of deformable objects from multiple demonstrations,” in *IEEE Int. Conf. Robot. Autom.*, 2015.

20. J. Van Den Berg, S. Miller, K. Goldberg, and P. Abbeel, "Gravity-based robotic cloth folding," in *Algorithmic Foundations of Robotics IX*, pp. 409–424, Springer, 2010.
21. K. Lakshmanan, A. Sachdev, Z. Xie, D. Berenson, K. Goldberg, and P. Abbeel, "A constraint-aware motion planning algorithm for robotic folding of clothes," in *Experimental Robotics*, pp. 547–562, 2013.
22. T. Tamei, T. Matsubara, A. Rai, and T. Shibata, "Reinforcement learning of clothing assistance with a dual-arm robot," in *IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 733–738, 2011.
23. N. Koganti, T. Tamei, K. Ikeda, and T. Shibata, "Bayesian nonparametric learning of cloth models for real-time state estimation," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 916–931, 2017.
24. A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Using depth and appearance features for informed robot grasping of highly wrinkled clothes," in *IEEE Int. Conf. Robot. Autom.*, pp. 1703–1708, 2012.
25. K. S. M. Sahari, H. Seki, Y. Kamiya, and M. Hikizu, "Edge tracing manipulation of clothes based on different gripper types," *J. Comput. Sci.*, vol. 6, no. 8, pp. 872–879, 2010.
26. F. Ferraguti, A. Pertosa, C. Secchi, C. Fantuzzi, and M. BonfÀ, "A methodology for comparative analysis of collaborative robots for industry 4.0," in *Design, Automation Test in Europe Conference Exhibition*, pp. 1070–1075, March 2019.
27. P. Chiacchio, S. Chiaverini, and B. Siciliano, "Direct and inverse kinematics for coordinated motion tasks of a two-manipulator system," *J. of dynamic systems, measurement, and control*, vol. 118, no. 4, pp. 691–697, 1996.

Paper D

Paper D

Textile Taxonomy and Classification Using Pulling and Twisting

Alberta Longhini, Michael C. Welle, Ioanna Mitsioni and Danica Kragic

Abstract

Identification of textile properties is an important milestone toward advanced robotic manipulation tasks that consider interaction with clothing items such as assisted dressing, laundry folding, automated sewing, textile recycling and reusing. Despite the abundance of work considering this class of deformable objects, many open problems remain. These relate to the choice and modelling of the sensory feedback as well as the control and planning of the interaction and manipulation strategies. Most importantly, there is no structured approach for studying and assessing different approaches that may bridge the gap between the robotics community and textile production industry. To this end, we outline a textile taxonomy considering fiber types and production methods, commonly used in textile industry. We devise datasets according to the taxonomy, and study how robotic actions, such as pulling and twisting of the textile samples, can be used for the classification. We also provide important insights from the perspective of visualization and interpretability of the gathered data.

1 Introduction

Interaction with deformable objects is an integral part of our everyday life but still a challenge for robotic systems. Work on robotic handling of textile or fabric traces back several decades [1] and, despite the clear need in industrial and domestic applications, many of the problems related to perception, planning and control remain open. From the industrial perspective, textile production and subsequent processes of garment design in fashion industry are largely not automated. Fashion industry is also undergoing an important transformation to address sustainability

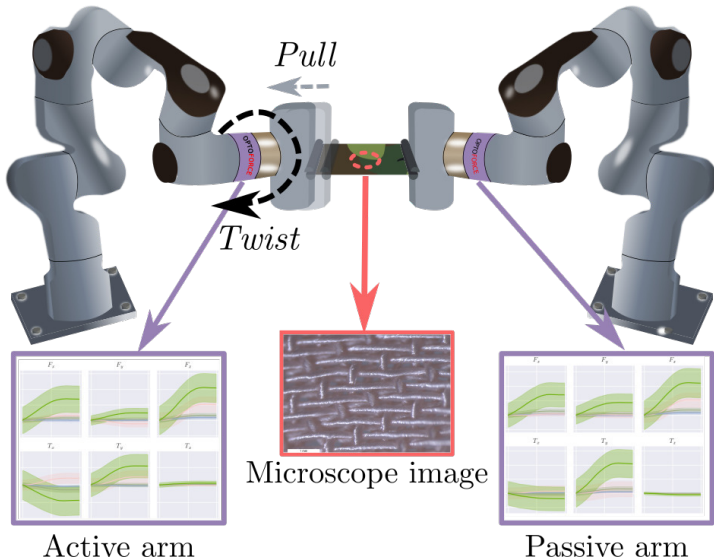


Figure D.1: System setup: Two 7 DoF Franka Emika Panda arms with force-torque sensors on the wrists, twisting and pulling a textile sample. Microscope images are used to define textile classes prior to training.

concerns, given that textile and clothing overproduction has a significant negative impact on the environment.

From the scientific perspective, robotic interaction with deformable materials has gained significant attention recently [2, 3]. Important milestones regarding the modelling, perception, planning, control and simulation of deformable materials have been identified but not yet reached. It may even be so that until robots reach the dexterity, flexibility and sensing that to some extent resembles human capabilities, successful interaction with deformable objects will remain a challenge. In robotics, textile has been used to study manipulation tasks like folding [4, 3, 5], robot-assisted dressing [6, 7, 8], garment recognition and classification [9, 10, 11, 12]. In most of these works, only a subset of textile properties is commonly considered, and textile is merely a tool for testing sensors [13] or control strategies [14].

In our work, we aim to study textile materials and their properties using physical interactions and wrist-mounted force-torque sensing. Similarly to humans, we aim at using actions such as pulling and twisting, to learn more about the textile properties, see Fig. D.1. The properties are defined using a textile taxonomy that follows the classification used in the textile production industry. Textile properties in general, and thus interaction dynamics, are affected by factors such as fiber material and production method - the fiber may be raw, coated or it may be a blend of several materials. Once used to produce garments or bed-clothing, the properties will change overtime based on washing, wearing, steaming - the textile can become

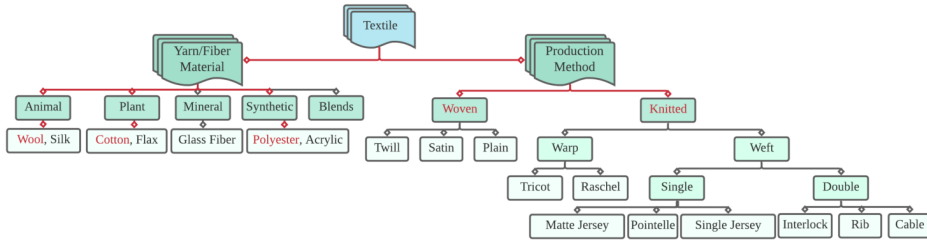


Figure D.2: Textile taxonomy considering yarn/fabric material and production method. Classes considered in this work are highlighted in red.

harder or softer, less or more elastic, thinner. The change in properties will also have an impact on the planning and control strategies used to interact with it - how we wash, iron and fold them, how we hold and manipulate garments when dressing somebody, whether we decide to recycle or reuse old garments.

The focus of this paper is to assess how a dual arm robotic system can be used to identify textile production methods through pulling and twisting. We propose to do so by learning a classifier on a dataset of textile samples that are annotated by their construction type, determined by inspecting their microscopic structure. We first outline a textile classification taxonomy related to both fiber type and production method, following notation used in textile industry. We then make a thorough study using a subset of materials and production techniques to assess the validity of our approach. We analyse two manipulation strategies as well as investigate which measurements are most relevant for classification. We conclude by discussing challenges and open problems.

2 Textile Taxonomy

Textile or fabric, is a deformable and flexible material made out of yarns or threads, which are put together by a construction or manufacturing process such as weaving, knitting, crocheting, knotting, tatting, felting, bonding or braiding. Most of the everyday clothing items we wear are constructed through weaving or knitting prior to sewing, although in high fashion other processes are used frequently too. Yarns and threads are produced by spinning raw fibers that may have different origin: animal, plant, mineral, synthetic or their blend. We summarize some of these aspects in Fig. D.2, showing also some of the further distinctions in terms of differences in the manufacturing process.

Woven fabric is usually produced by using two sets of yarn, while knitted fabric employs a single set. To produce woven fabric, the yarn is interlaced, as opposed to knitting where it is interloped. Due to its construction, woven fabric is often hard and nonelastic, allowing it to hold creases well, and can be only stretched diagonally if the yarn itself is not a blend that includes elastic material. It is commonly used

D4

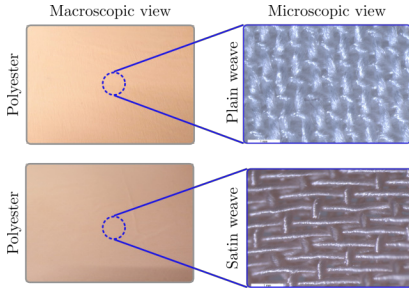


Figure D.3: Textile can look the same given regular camera (left), but very different on the microscopic level (right). The different weaving styles (plain weave and satin weave) with the same material determine the dynamical properties that are important when manipulating fabric.

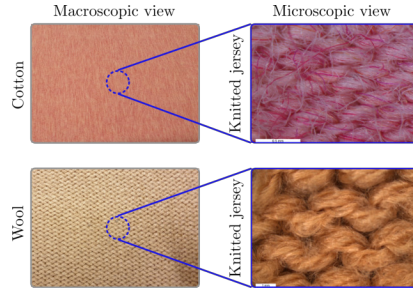


Figure D.4: Textile can look different on the macro level, but very similar on the micro level. Left) two different materials (Wool and Cotton) using a regular camera. The microscope image (right) reveals that they have the same underlying construction - knitted jersey.

to produce garments such as shirts and jeans. On the other hand, knitted fabric is soft and can be stretched in all directions, making it rather wrinkle-resistant. One example of its frequent use is for t-shirts. One important aspect is that it usually does not stretch equally in all directions - for example, a t-shirt will stretch more horizontally than vertically to more naturally follow body shape. A more complete account of the properties of fabric can be found in [15].

The above is of importance for various robotics applications that consider active interaction with the textile. For assisted dressing applications, it is important for the robotic system to generate relevant control strategies when pulling up pants, helping with the sleeves or pulling down the t-shirt: hard textile may require completely different manipulation strategies and safety considerations than the flexible one. Most of the clothing items will have a content label attached to them and may offer information about the type of yarn used. However, the manufacturing process is never described on the label, neither is the fact that a clothing item may be a combination of woven and knitted parts and a combination of different type of textiles being put together in the sewing step. For example, both a pair of jeans and a t-shirt may be made with 100% cotton textile that, in the first case, is woven and in the second case, knitted. With the proposed taxonomy and the work in this paper, we report on some initial insights on how some of the textile properties can be examined by using force-torque measurements and actions such as pulling and twisting of the textile samples.

Even for humans, the manufacturing method may not be visible with the naked eye and the label only provides the yarn material. We use our experience of previously interacting with clothing items to choose appropriate actions when dressing

ourselves or others, washing, ironing, repairing or sewing. To shed some light on this, we collected microscope images of our textile samples. While a regular camera image may not provide enough signal resolution in actual robot interaction with the textile, the high-resolution microscope images show different ways of interlacing yarn that has a huge effect on the elasticity of the textile Fig. D.3. Similarly, garments with different reflection or texture properties may look rather different under a regular camera but their dynamical properties may be the same if the manufacturing method and yarn type are the same, see Fig. D.4.

In this paper, we therefore set out to investigate how force-torque measurements together with actions such as pulling and twisting may be used to classify the textiles according to the proposed taxonomy. We chose pulling and twisting since these are also the two most common actions humans use for inspecting textile properties.

In robotics, textile materials have been considered from perception, learning, planning and control perspectives. Most notable applications consider folding, assisted dressing or material classification [16, 2].

Despite the broad interest in the computer vision community, most works concentrate on building clothing item taxonomies [9] rather than identifying material properties. Problems such as garment motion prediction [17], classification [10], dressing 3D simulated humans [18], have also been addressed. It has also been shown that wrinkle detection may be helpful for classification [19, 20]. However, with vision alone it may be difficult to estimate the physical attributes of textiles [21] although the results in [22] indicate that vibrations captured in video can be correlated to the stiffness and density of fabrics.

In robotics, identifying textile properties is important, but there is no common taxonomy that allows for comparison and benchmarking of the proposed approaches. Recent work in [23] proposes a taxonomy of 184 materials including leather, fur and plant fiber but there is no focus on textile in particular or the production method. Haptic feedback has often been used to label various types of materials [24, 13, 25]. The authors in [26] study compliance and texture to classify 32 materials including textile. Non-contact techniques have been used in [27] to distinguish among five material categories, one of which was textile. Thus, none of these works focuses specifically on textile material, or considers fiber and production method in particular. When considering textile classification, it has been studied from the fiber material perspective [28, 7] or properties such as thickness, softness and durability [29]. Material texture identification has been addressed in [30, 31, 32], without considering the difference between fiber material and textile production method. Given these, we believe that our initial study and outlined taxonomy provides examples of how textile classification can be studied in a more structured manner.

3 Data collection and dataset design

For this initial study, we relied on 40 textile samples, 10 each of polyester and wool, and 20 cotton samples. From the samples, we cut out pieces 40×17 cm in size.

Polyester samples are woven and wool samples are knitted. Out of the 20 cotton samples, 10 are woven and 10 are knitted. We cut the pieces so that the yarn direction is along the axis of pulling, with the more elastic direction of stretching being orthogonal to the axis of pulling as can be seen in Fig. D.5. Two Franka Emika Panda arms are equipped with wrist-mounted Optoforce 6-axis Force-Torque (FT) sensors and flat 3D-printed grippers.

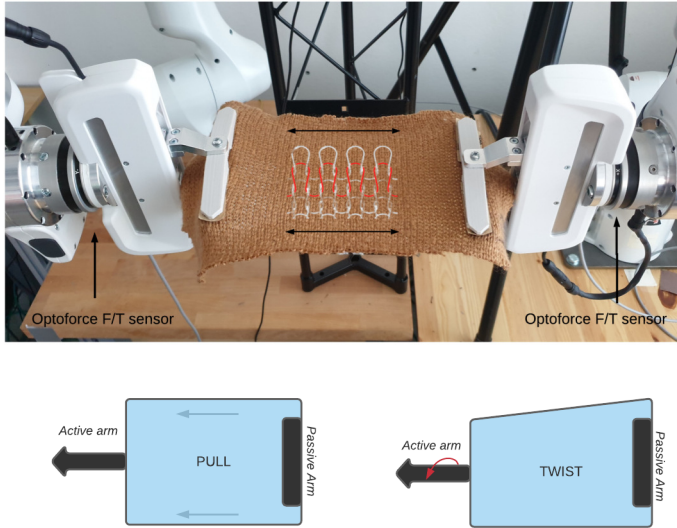


Figure D.5: (Top) Flat 3D-printed grippers holding wool sample. The yarn direction coincides with the pulling direction. FT sensors are mounted on the wrist of the manipulators. (Bottom) Schematic example of *pull* and *twist*.

For data collection, we aimed to capture the samples' properties by means of two exploratory procedures [33], *pulling* and *twisting*, and investigate if they are consistently classifiable. To further analyse different data collection strategies, we decided to let just one arm move (also called *active arm*) while the other one is kept still (*passive arm*), see Fig. D.5. A precise definition of these two manipulation actions is the following:

- **Pull** The active arm exerts force on the sample by steadily moving 2 cm away from the static passive arm, maintaining a motion direction parallel to the grasping plane.
- **Twist** The active arm's end-effector rotates 90 degrees, while the arm is pulling to ensure the sample is stretched adequately to capture its reaction to torsion.

Each textile sample was held with a grasping force of 20N by both robot arms and pulled and twisted 20 times. As each sequence of pulling and twisting may result in a slight offset of the contact point, we re-positioned the sample in the hand to the original starting points after every five pulls/twists. Force and torque signals were recorded for a duration of 2s from each sensor at a frequency of 1kHz. Thus, for each textile sample, we have 20 examples, one for each arm, with $2 \times 1000 \times 6$ raw FT measurements.

3.1 Dataset design

We first sub-sampled the raw measurements for each FT dimension. We performed average downsampling to 150 values as a trade-off between noise reduction and information loss. Given these, we build 6 datasets:

- $\mathcal{D}_{active}^{twist}$, $\mathcal{D}_{active}^{pull}$, $\mathcal{D}_{passive}^{twist}$, $\mathcal{D}_{passive}^{pull}$: 4 datasets corresponding to the two actions for each arm individually.
- \mathcal{D}^{twist} and \mathcal{D}^{pull} : 2 datasets corresponding to the two actions and the integrated measurements from the two arms.

In summary, the datasets were labelled to represent the samples of the taxonomy in Fig. D.2 highlighted in red. Therefore, by considering the "Production Method" branch we obtained 2 classes: woven and knitted, while on the "Yarn/Fiber Material" branch, we obtained 3 classes: wool, polyester and cotton.

4 Data Visualization and Dataset Insights

We first inspect the generated datasets to assess to what extent the collected data, labelled according to the proposed taxonomy, are representative for classification. To this end, we employ t-SNE[34] and project the datasets into a two-dimensional space. More specifically, we want to answer the following questions:

- Can the generated data and employed actions show a clear distinction between woven and knitted textiles?
- Is there a difference between pulling and twisting, in terms of how informative they are, for the woven vs knitted classification?
- What is the effect of fiber type on the classification performance and can we distinguish not only the production method, but also the fiber type given our datasets?

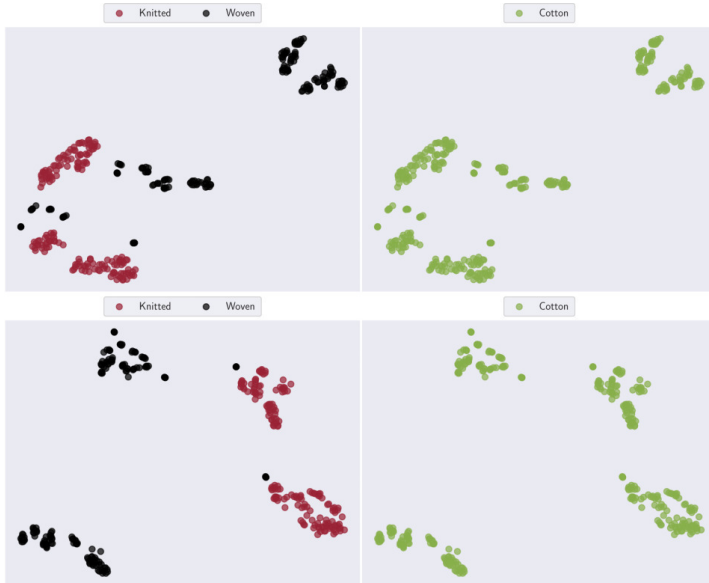


Figure D.6: Effect of splitting construction method on Cotton samples. t-SNE plot of measurements obtained from the active (top row) and passive (bottom row) arms during pulling Cotton samples.

4.1 Insight 1: Production Method

First, we visually inspect if the datasets are representative of the production method. Fig.D.6 shows the distribution of the data projected in 2D when only cotton is considered, where the left side of the figures shows how the data are separated by the production method, woven (also called Cotton-Twill) and knitted (also called Cotton-Jersey). The top row corresponds to the active arm and the bottom row to the passive arm during a pulling trial. From the figure we observe that by considering the actual production method we obtain clearly structured groups of samples that would have been otherwise masked by categorizing them as the same material.

4.2 Insight 2: Pulling vs Twisting

Second, we assess whether there is an advantage in using both pulling and twisting. As a first step, Fig. D.7 shows that measurements for the two actions exhibit different behaviors. For example, the force measurements during pulling exhibit more variety than they do for twisting, indicating that they may carry more information for the different classes.

To further examine how important the different measurements are and how they can affect classification, we train a simple SVM [35] classifier to predict the sample's class, while trained on individual signals $(F_x, F_y, F_z, T_x, T_y, T_z)$. The classifier

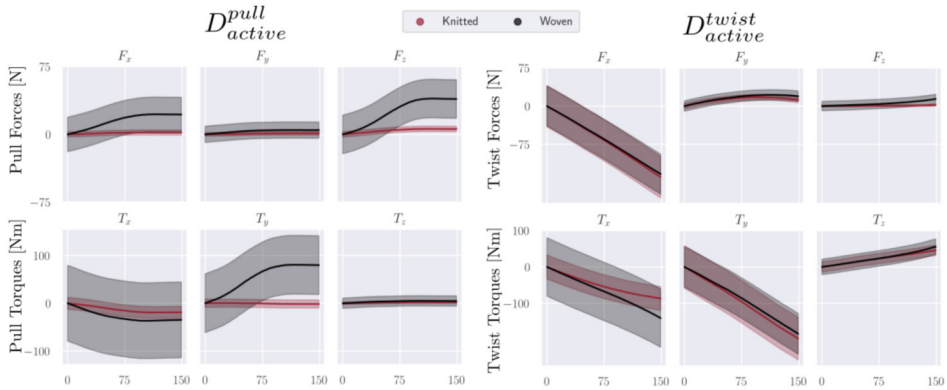


Figure D.7: Pull vs twist measurements on the active arm.

has a linear kernel and it is implemented with Scikit-learn [36]. Table D.1 shows the test set accuracy based on the individual signals for the *pull* (left) and the *twist* (right) strategies of the *active* arm, when learning on a material-based dataset and a construction-based ones.

Signal	<i>Pull</i>		<i>Twist</i>	
	Material	Construction	Material	Construction
F_x	30%	82%	45%	57%
F_y	32%	63%	38%	52%
F_z	70%	100%	53%	73%
T_x	42%	75%	38%	63%
T_y	44%	83%	38%	63%
T_z	40%	53%	45%	59%
All	80%	100%	70%	87%

Table D.1: SVM test set performance based on the individual signals for a dataset with 3 classes for material distinction and 2 classes for the production methods.

From the SVM results, we observe that for the material-based classification, the accuracy scores for the two actions are comparable and rather low. However, when considering the proposed labels, the accuracy increases and in almost all signal cases, pulling outperforms twisting.

These results reinforce that following construction-based taxonomy is advantageous as well as the notion that the sensory feedback varies a lot depending on how

textile is manipulated. It is therefore of fundamental importance to understand how to choose the proper exploration strategy. Moreover, as mentioned in Section 2, the way in which textile threads are interlocked leads to different elastic properties. Knitted textiles for example, can be stretched lengthwise or along the horizontal direction. Woven textiles instead, are usually not stretchable, apart from a bias direction that for example, for denim is the diagonal one. All these concepts play an important role in classification and highly increase the complexity of the task.

4.3 Insight 3: Fiber Material vs Production Method

Lastly, we assess to what extent the fiber type can be identified in addition to the production method. An example of this can be seen in Fig. D.8 for dataset $\mathcal{D}_{passive}^{twist}$ that depicts the difference between samples categorized using just their production method and sample categorized using both their material and production method. More specifically, the figure portrays the differences between a simple categorization of knitted/woven and a further distinction of the Cotton class, which is split into Cotton-Twill (woven) and Cotton-Jersey (knitted). We can see that Cotton-Twill visually belongs to a separate cluster as observed in section 4.1. It can be also noticed that some of the Cotton-Twill samples are closer to Polyester as to Wool, while Cotton-Jersey is closer to Wool than to Polyester.

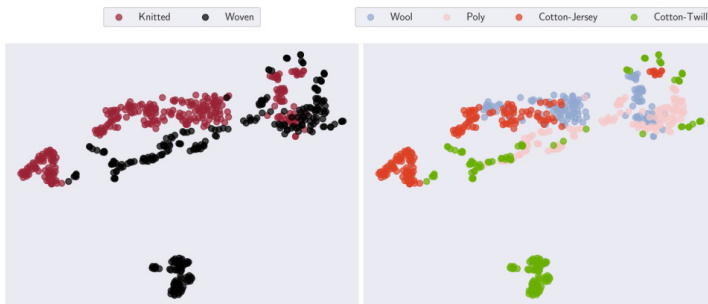


Figure D.8: Twist on the passive arm: visualization with respect to (left) production method, (right) fiber material.

We can also observe the effect of the proposed taxonomy on the individual signal level, by considering the dataset $\mathcal{D}_{active}^{pull}$ for example. Fig. D.9 depicts how the split of Cotton by construction method highlights the difference between the mean force used for Cotton-Twill and Cotton-Jersey at the end of the pulling action, further showing the necessity of splitting the Cotton class. Moreover, besides the detectable distinction among Polyester, Wool, Cotton-Twill and Cotton-Jersey signals, woven materials keep being the ones with higher mean force while knitted ones are in general less tension-resistant, reflecting the behaviour of the construction methods.

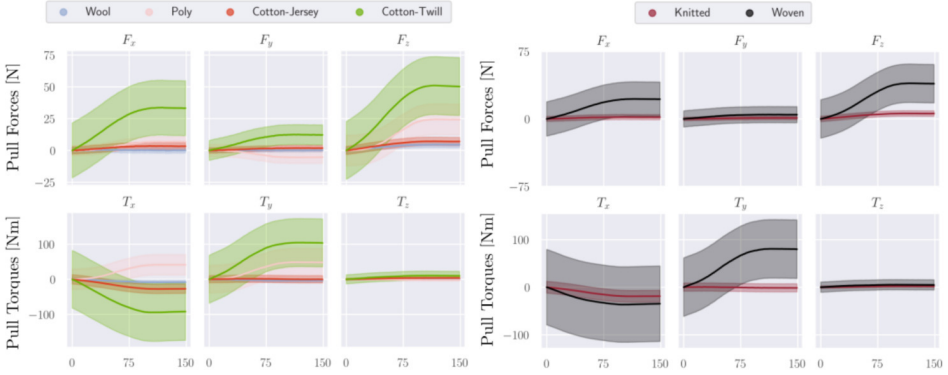


Figure D.9: Mean and standard deviation of the measurements sensed by the active arm while pulling.

5 Classification and Interpretability

The next step is to assess the classification performance using a more complex architecture, like a CNN model. The input of the network are vectors of the six concatenated FT measurements, the network consists of four 2D convolutional layers and its activations are rectified linear units (ReLU). Furthermore, we adopt rectangular kernels of size 5×1 which convolve across measurements of the same signal as done in [37]. The output sizes of the convolutional layers are respectively 24, 12, 8 and 4.

The features learned from the last convolutional layer are flattened and fed to a fully connected layer with 48 hidden neurons. The outputs of this block are the predicted class probabilities. We also consider the case of the joined measurements for the *active* and *passive* arm, using the same architecture but adjusting the size of the fully-connected layer to accommodate the 150×12 signal.

We partitioned each dataset into 90/10 train/test splits. Table D.2 summarizes the classification results. Firstly, we observe that the construction-based labeling outperforms the material-based one for any dataset or action. More specifically, using our taxonomy both actions provide enough information for accurate predictions. However, twisting is consistently less accurate than pulling for all labeling and datasets considered.

These observations are validated in the case of the joined datasets with \mathcal{D}^{pull} achieving excellent performance in distinguishing the construction method of the textile, leading to the conclusion that pulling is a better option for classification.

Input Dataset	Materials	Construction
$\mathcal{D}_{active}^{pull}$	87.5%	100%
$\mathcal{D}_{passive}^{pull}$	85.8%	96.7%
$\mathcal{D}_{active}^{twist}$	76.7%	95.0%
$\mathcal{D}_{passive}^{twist}$	78.3%	89.2%
\mathcal{D}^{pull}	95.0%	100%
\mathcal{D}^{twist}	79.0%	91.7%

Table D.2: Test accuracy with a CNN model for all the different datasets when following the material-based labelling and the proposed, construction-based one.

5.1 Interpretability and measurement assessment

To further examine the effect of the different measurements for classification, we interpret the results from the CNN model through GradCAM [38]. GradCAM is an interpretability technique that produces visual explanations in the form of heatmaps that portray which parts of the input contribute the most to the predicted label. We follow the same methodology as in [39] to produce and inspect the contribution of each feature in samples from datasets \mathcal{D}^{pull} and $\mathcal{D}_{active}^{pull}$ of Table D.2.

An example of the heatmaps can be seen in Fig. D.10 for two correctly classified samples of woven cotton and knitted cotton from the dataset \mathcal{D}^{pull} . Every row corresponds to a different measurement channel and its color is defined by how important it is for the prediction. The importance is scaled between 0 and 1 and follows the colormap on the right of the images. Fig. D.10 shows that for both cotton instances, the network is focusing on the same features between the passive and the active arm. Concretely, for woven cotton, the forces on both axes Z are the most important features, followed by the torques T_x^p, T_x^a and some parts of T_y^p, T_y^a . However, for knitted cotton, the network utilized all the force measurements for both arms and the torque on axis Z for the active arm. These results indicate that even when utilizing the material based-labeling, the CNN network focuses on different patterns when classifying samples of the same material but different construction methods.

Finally, we inspect two classification results from dataset $\mathcal{D}_{active}^{pull}$ when it is labelled according to the proposed taxonomy. The left heatmap corresponds to a correct classification of a woven sample and the right heatmap on the correct classification of a knitted one. The important features agree with the intuition

gained from Table D.1 as the decisions are heavily based on the ones highlighted also by the SVM, namely forces F_z and torques T_x, T_y .

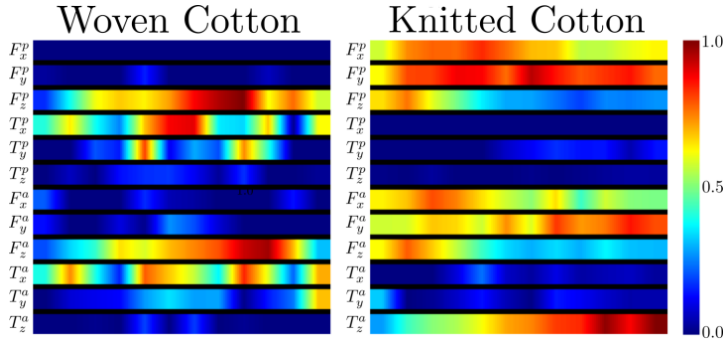


Figure D.10: Heatmaps of feature importance for the classification using dataset \mathcal{D}^{pull} . The intensity of each row (with red being the most important) denotes what the network focuses on to classify the sample.

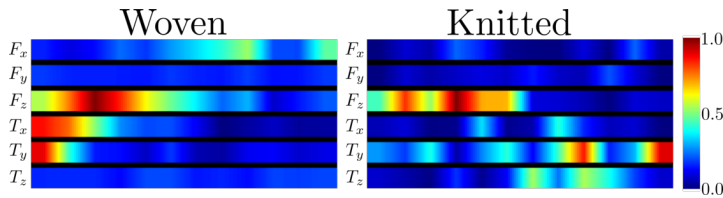


Figure D.11: Heatmaps for the dataset $\mathcal{D}_{active}^{pull}$: activations for a correctly classified Woven sample (left) and a correctly classified Knitted sample (right) correctly classified Knitted sample.

We note that the visualization shows only where the neurons of the network are most active for single examples. It is possible for a network to construct multiple patterns to classify the same class, making generalisation difficult. We can however, observe that certain measurements are more important for classification than others, which is a valuable insight when designing future active exploration strategies.

6 Discussion and Conclusion

In this work, we outlined a textile taxonomy and showed our initial results on textile sample classification using pulling and twisting actions. The focus of the study was to assess to what extent a taxonomy used in textile industry is a viable model to structure robotic interaction and provide a basis for a whole new area of structured studies of this class of deformable materials.

We provided insights on how a combination of different actions and FT measurements vary with respect to textile production method and fiber material. Pulling and twisting, as inspired by the human interaction with textile, are viable choices of actions and they provide relevant information for classification. One interesting question that arises is what other actions can be potentially employed and to what extent dexterous hand/finger motion could be exploited in addition to pulling and twisting. Other works in literature demonstrated the use of specialized fingers and sensors for this purpose and it is yet to be seen to what extent we can consider such solutions to become commercial.

Combining multiple actions, as well as passive and active interaction, is also an interesting aspect to be explored. We may start with pulling/stretching and based on the first step classification, subsequent routines may be performed more suitably for identifying categories of interest, such as for example fiber material, elasticity, whether the textile is wet or dry, etc. Here, reinforcement learning may be used to learn actions that maximize the utility of the sensor readings for discriminating various textile properties.

We also performed an initial study using visual feedback under pulling and twisting. However, for the considered categories, regular cameras do not provide enough resolution to bring sufficient information on the production method or the fibre type. One could potentially rely on the reflectance properties of textile materials, but most of the works in this area that stem from the computer vision community, are not applicable in uncontrolled settings that would occur in real-life applications. An idea supported by our experimental/empirical observations was the fact that creases and wrinkles on the textile fabric may be a useful feature to exploit for certain applications. When pulling or stretching the fabric in many different directions, creases and wrinkles will vary dependent on the properties of the textile: dense and hard textile creases differently from soft and thin textile. In such cases, integrating vision and FT may be useful. Careful consideration on what visual features are used needs to be taken into account. For example, using flow-based methods [40] or specified wrinkle detectors [41] dealing with various texture properties may be considered.

An additional important aspect to be considered is the ability to assess how textile properties change over time. Certain textiles are made to be more durable, fibers are blended, their use and handling in terms of washing, ironing, folding, will affect how clothing items deteriorate over time. In other words, the information of the fiber content usually available on the label sewn on the clothing item, may be helpful but it is not fully relevant. For example, a T-shirt made out of cotton, may be more elastic and thicker when new, and rather thin and almost non-elastic after many washings. Thus, its handling in terms of washing and ironing will be different, as well as one may decide to keep or reuse a newer one, and recycle a well-used one.

The outlined taxonomy, visualization, CNN classification and measurement interpretability are important tools that can provide more insight into the difficulty of the considered problem. The taxonomy provides a structured approach to study

textile materials and has not been previously considered in the area of robotics. We also need an approach that brings the robotics community closer to textile production industry and this is one way of achieving that. We provided several examples of how the generated textile material classes are a viable approach and how these can be studied together with actions such as pulling and twisting.

Initial classification results using deep neural networks show a good potential and we will build on these with a more extensive database of samples, actions and multimodal sensory feedback. More specifically, we will study a richer set of pulling actions, with samples of different sizes also considering standardized textile for the purpose of repeatability, reproducibility and replicability. We believe that this study is an important step toward a more robust and versatile textile handling and manipulation for applications such as various household tasks, assisted dressing and recycling.

7 Acknowledgements

This work has been supported by the European Research Council, Swedish Research Council and Knut and Alice Wallenberg Foundation.

References

1. K. Paraschidis, N. Fahantidis, V.-Petridis, Z. Doulgeri, L. Petrou, and G. Hasapis, “A robotic system for handling textile and non rigid flat materials,” *Computers in Industry*, vol. 26, no. 3, pp. 303–313, 1995, computer Integrated Manufacturing and Industrial Automation.
2. J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey,” *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.
3. A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. PetrÅk, A. Kargakos, L. Wagner, V. Hlaváč, T. Kim, and S. Malassiotis, “Folding clothes autonomously: A complete pipeline,” *IEEE Transactions on Robotics*, vol. 32, no. 7, pp. 688–716, 2018.
4. Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, “Folding deformable objects using predictive simulation and trajectory optimization,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 6000–6006.
5. 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,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.
6. A. Kapusta, W. Yu, T. Bhattacharjee, C. K. Liu, G. Turk, and C. C. Kemp, “Data-driven haptic perception for robot-assisted dressing,” in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016, pp. 451–458.

7. G. Chance, A. Jevtić, P. Caleb-Solly, and S. Dogramadzi, “A quantitative analysis of dressing dynamics for robotic dressing assistance,” *Frontiers in Robotics and AI*, vol. 4, pp. 1–14, 05 2017.
8. I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Alenya *et al.*, “Benchmarking bimanual cloth manipulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1111–1118, 2020.
9. Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, “Deepfashion: Powering robust clothes recognition and retrieval with rich annotations,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1096–1104.
10. J. Huang, R. S. Feris, Q. Chen, and S. Yan, “Cross-domain image retrieval with a dual attribute-aware ranking network,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1062–1070.
11. Y. Ge, R. Zhang, X. Wang, X. Tang, and P. Luo, “Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5337–5345.
12. T. Ziegler, J. Butepage, M. C. Welle, A. Varava, T. Novkovic, and D. Kragic, “Fashion landmark detection and category classification for robotics,” in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2020, pp. 81–88.
13. E. Kerr, T. McGinnity, and S. Coleman, “Material recognition using tactile sensing,” *Expert Systems with Applications*, vol. 94, pp. 94–111, 2018.
14. V. Petrik and V. Kyrki, “Feedback-based fabric strip folding,” *arXiv preprint arXiv:1904.01298*, 2019.
15. S. Grishanov, “2 - structure and properties of textile materials,” in *Handbook of Textile and Industrial Dyeing*, ser. Woodhead Publishing Series in Textiles, M. Clark, Ed. Woodhead Publishing, 2011, vol. 1, pp. 28–63.
16. T. C. JimÁñez P., “Perception of cloth in assistive robotic manipulation tasks,” vol. 19, pp. 409–431, 2020.
17. C. Patel, Z. Liao, and G. Pons-Moll, “Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
18. B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll, “Multi-garment net: Learning to dress 3d people from images,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
19. L. Sun, S. Rogers, G. Aragon-Camarasa, and J. P. Siebert, “Recognising the clothing categories from free-configuration using gaussian-process-based interactive perception,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2464–2470.

20. J. Silvestre-Blanes, J. Berenguer-Sebastian, R. Perez-Llorens, I. Miralles, and J. Moreno, “Garment smoothness appearance evaluation through computer vision,” *Textile Research Journal*, vol. 82, no. 3, pp. 299–309, 2012.
21. S. Luo, J. Bimbo, R. Dahiya, and H. Liu, “Robotic tactile perception of object properties: A review,” *ArXiv*, vol. abs/1711.03810, 2017.
22. A. Davis, K. L. Bouman, J. G. Chen, M. Rubinstein, F. Durand, and W. T. Freeman, “Visual vibrometry: Estimating material properties from small motion in video,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
23. M. Strese, L. Brudermueller, J. Kirsch, and E. Steinbach, “Haptic material analysis and classification inspired by human exploratory procedures,” *IEEE Transactions on Haptics*, vol. 13, pp. 404–424, 2020.
24. A. J. Fishel and E. G. Loeb, “Bayesian exploration for intelligent identification of textures,” *Front. Neurobot.*, 2012.
25. J. M. Romano and K. J. Kuchenbecker, “Methods for robotic tool-mediated haptic surface recognition,” in *2014 IEEE Haptics Symposium (HAPTICS)*, 2014, pp. 49–56.
26. M. Kerzel, M. Ali, H. G. Ng, and S. Wermter, “Haptic material classification with a multi-channel neural network,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 439–446.
27. Z. Erickson, E. Xing, B. Srirangam, S. Chernova, and C. Kemp, “Multimodal material classification for robots using spectroscopy and high resolution texture imaging,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.
28. A. A. Khan, M. Khosravi, S. Denei, P. Maiolino, W. Kasprzak, F. Mastrogiovanni, and G. Cannata, “A tactile-based fabric learning and classification architecture,” in *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, 2016, pp. 1–6.
29. W. Yuan, Y. Mo, S. Wang, and E. Adelson, “Active clothing material perception using tactile sensing and deep learning,” 05 2018, pp. 1–8.
30. J. Sinapov, V. Sukhoy, R. Sahai, and A. Stoytchev, “Vibrotactile recognition and categorization of surfaces by a humanoid robot,” *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 488–497, 2011.
31. F. V. Drigalski, M. Gall, S.-G. Cho, M. Ding, J. Takamatsu, T. Ogasawara, and T. Asfour, “Textile identification using fingertip motion and 3d force sensors in an open-source gripper,” *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 424–429, 2017.
32. S. Luo, W. Yuan, E. Adelson, A. Cohn, and R. Fuentes, “Vitic: Feature sharing between vision and tactile sensing for cloth texture recognition,” *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2722–2727, 2018.

33. S. J. Lederman and R. L. Klatzky, "Haptic classification of common objects: Knowledge-driven exploration," *Cognitive Psychology*, vol. 22, no. 4, pp. 421–459, 1990.
34. L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
35. C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
36. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
37. Y. Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell, "Deep learning for tactile understanding from visual and haptic data," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 536–543.
38. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct 2019.
39. I. Mitsioni, J. MÅnttÅri, Y. Karayiannidis, J. Folkesson, and D. Kragic, "Interpretability in contact-rich manipulation via kinodynamic images," 2021.
40. V. Scholz and M. A. Magnor, "Cloth motion from optical flow." in *VMV*, vol. 4, 2004, pp. 117–124.
41. M. Takamatsu, E. Iwata, T. Furukawa, and M. Hasegawa, "A study on garment wrinkle detection through a 3d camera scanning with normal map and hough transform," in *International Workshop on Advanced Image Technology (IWAIT) 2019*, vol. 11049. International Society for Optics and Photonics, 2019, p. 110492U.

Paper E

Paper E

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

Constantinos Chamzas*, Martina Lippi*, Michael C. Welle*, Anastasia Varava, Lydia E. Kavraki, and Danica Kragic

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.

1 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

* Authors contributed equally, listed in alphabetical order.

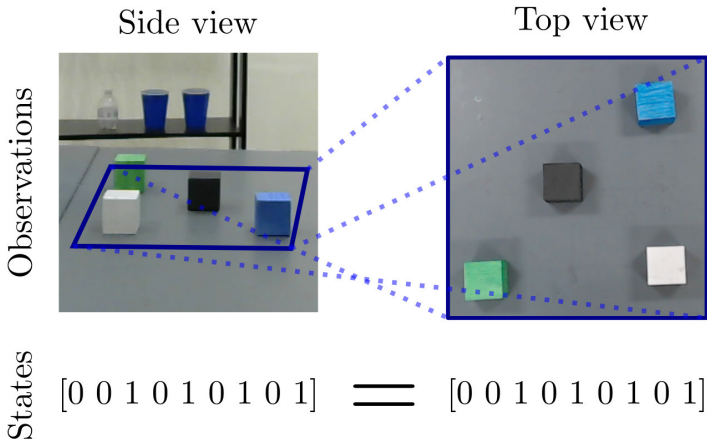


Figure E.1: Examples of visually different observations (different views) of the same state (arrangement of the boxes).

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 Figure E.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.

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 models¹.

2 Related Work

State representation learning from high-dimensional data has been successfully used in a variety of robotic tasks. As shown in Table E.1, 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.

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

Table E.1: Overview of loss functions (reconstruction, KL divergence and contrastive) used by state-of-the-art methods.

As shown in Table E.1, 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

¹ <https://state-representation.github.io/web/>

E4

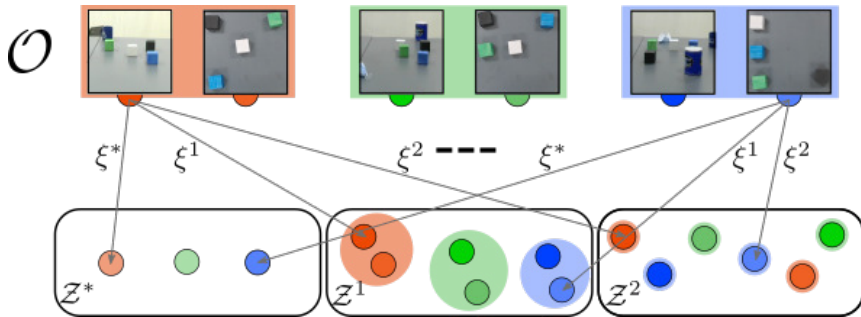


Figure E.2: Examples of mapping functions ξ^* , ξ^1 , ξ^2 (arrows) from observation space \mathcal{O} (top row) to latent spaces \mathcal{Z}^* , \mathcal{Z}^1 , \mathcal{Z}^2 (bottom row). Boxes arrangement represents the system state and images marked with variations of the same color contain the same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

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 section 3.

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.

3 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 Figure E.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 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 Figure E.3 for a box manipulation task (with interchangeable

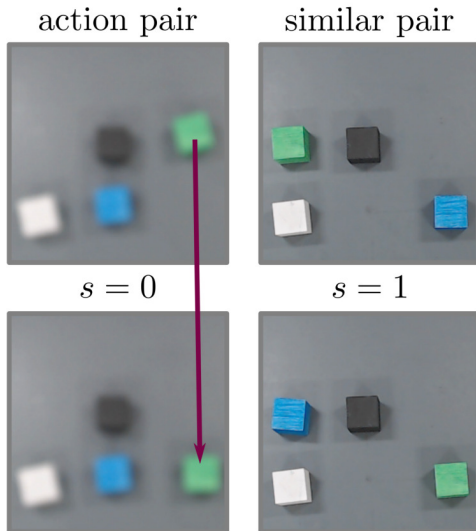


Figure E.3: Example of action (left) and similar (right) pairs. We consider the boxes interchangeable (only the resulting arrangement matters).

boxes), as detailed in section 7. 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 interchangeable 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 $\{o_1^s, \dots, o_n^s\} \subset \mathcal{O}_T$ in the dataset and define the tuples $(o_i, o_k^s, s = 0)$, $k = 1, \dots, n$, 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 $\xi^*(o_i) = \xi^*(o_k^s)$.

4 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:

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

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]:

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

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

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

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 following loss function [23]

$$\begin{aligned} \mathcal{L}_{pc\text{-vae}}(o_i, o_j, s) &= \frac{1}{2} (\mathcal{L}_{\beta\text{-vae}}(o_i) + \mathcal{L}_{\beta\text{-vae}}(o_j)) \\ &\quad + \gamma \mathcal{L}_{pc}(o_i, o_j, s) \end{aligned}$$

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

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

Table E.2: Summary of the considered models with respect to their loss functions.

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 (E.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]:

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

where $\text{sim}(u, v) = u^\top v / \|u\| \|v\|$ 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 (E.2).

Models Summary: As summarized in Table E.2, 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 our experiments in section 8, we also compare to the case in which no model is used, and raw observations are directly exploited.

5 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 4 shows a high level description of the LSR building procedure.

Algorithm 4 Adapted LSR building [23]

Require: Dataset \mathcal{T}_z , min cluster size m

$\mathcal{G} = \text{build-reference-graph}(\mathcal{T}_z)$	# Phase 1
$\mathcal{C}_z = \text{HDBSCAN-clustering}(\mathcal{T}_z, m)$	# Phase 2
$\text{LSR} = \text{build-LSR}(\mathcal{G}, \mathcal{C}_z)$	# 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 section 4.

6 Representation Evaluation Metrics

To evaluate the performance of the different latent mapping models, we propose two types of metrics. First, as stated in section 3, 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.

6.1 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 result in $h_c = 1$ and $c_c = 0$, while assigning all elements in the same cluster would result 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.

6.2 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*.

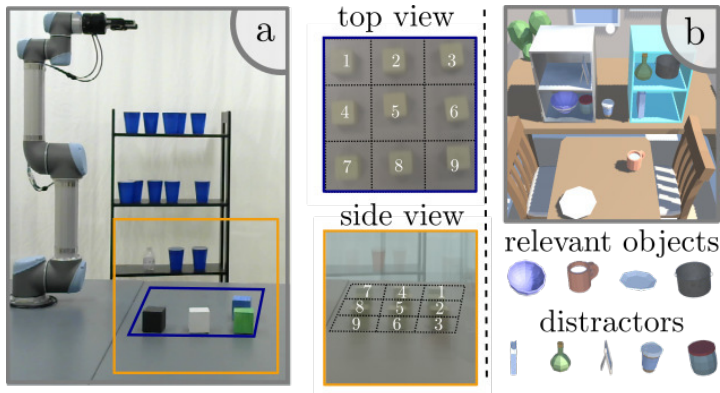


Figure E.4: **a)** Box manipulation dataset with two viewpoints. **b)** Shelf arrangement dataset with the task relevant objects (top object row) and the five distractor objects (bottom object row).

7 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] as well as in Appendix 10.1. 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 state 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 website¹.

Box Manipulation: The setup of this real-world case study is shown in Figure E.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 Figure E.4), *ii)* \mathcal{BM}_t , where the observations are only taken from the top view (in blue in Figure E.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 back-

ground (side view), as well as out-of-focus images. In Figure E.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-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 Figure E.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. E.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). For more information about the tasks and their rules see Appendix 10.2-10.3.

We trained each of the seven models in Sec. 4 (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. 3. 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 in the Appendix 10.4, the website¹, and the code².

8 Results and Discussion

Two main questions are discussed in detail:

²<https://github.com/State-Representation/code>

Models	Dataset \mathcal{BM}_t							
	\mathcal{V}	h_c	c_c	s_c	\mathcal{E}	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
	Dataset \mathcal{BM}_{st}							
-	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 E.3: Evaluation results for the latent mapping models and raw observations on \mathcal{BM}_t (top row) and \mathcal{BM}_{st} (third row) and their augmented versions (second and fourth row respectively) $\overline{\mathcal{BM}}_t$ and $\overline{\mathcal{BM}}_{st}$ for the box manipulation task. Best results in bold.

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. 3 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 E.3 (top and third row). 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

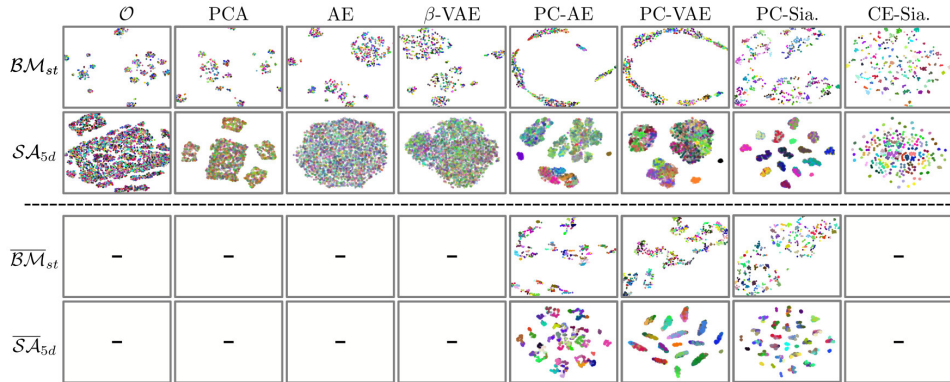


Figure E.5: Two-dimensional t-SNE plots for the box manipulation task for the mixed view (top row) and the shelf arrangement task for the distractor case (second row). Each color is associated with a possible underlying state. On the third and fourth row we display the plots for the augmented contrastive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

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 $\mathcal{B}\mathcal{M}_{st}$ (third row), 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. Results with $\mathcal{B}\mathcal{M}_s$ can be found in Appendix 10.5.

The above results are also reflected in the qualitative analysis in Figure E.5 (top two rows) 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 $\mathcal{B}\mathcal{M}_{st}$ are shown. We can notice that the raw observations, PCA and purely reconstruction-based models AE and β -VAE fail in structuring the representations, forming spurious clusters in which different states are mixed up. Non-homogeneous

Models	Dataset \mathcal{SA}_{0d}							
	$ \mathcal{V} $	h_c	c_c	s_c	$ \mathcal{E} $	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
	Dataset \mathcal{SA}_{5d}							
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 E.4: Evaluation results for the contrastive-based latent mapping models on \mathcal{SA}_{0d} (top row) and \mathcal{SA}_{5d} (third row) and their augmented versions $\overline{\mathcal{SA}}_{0d}$ (second row) and $\overline{\mathcal{SA}}_{5d}$ (forth row) for the shelf arrangement task. Best results in bold.

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 shallower² 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. 3, 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. Full results can be found in Appendix 10.6. Table E.4 reports the obtained evaluation metrics. When no augmentation is applied (top and third row), 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 (second row),

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 (forth row). 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 E.3, where we can notice that, when the augmentation is used (second and forth row), 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.

Figure E.5 reports the t-SNE plots for the shelf stacking task with five distractors (\mathcal{SA}_{5d}) obtained with (forth row) and without (second row) 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 and third row of Figure E.5). *In summary, we observe that a simple data augmentation boosts the performance of the contrastive-based models.*

9 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.

10 Appendix

10.1 Simulated Box Stacking Task

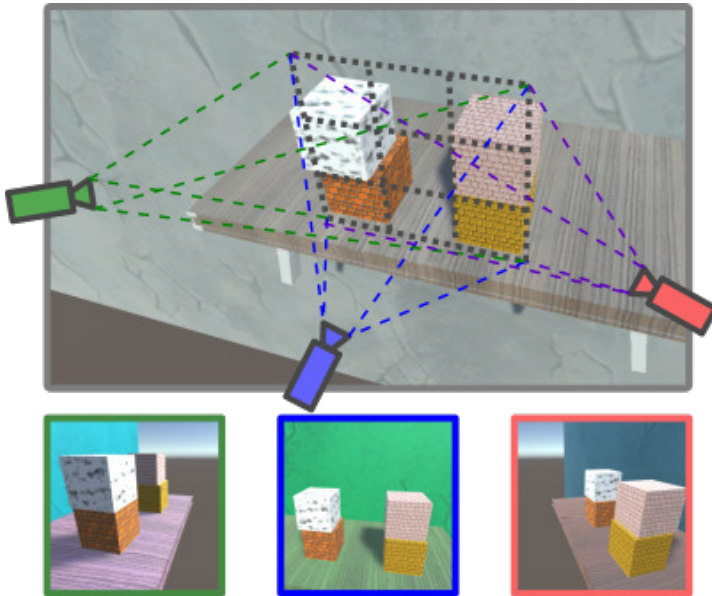


Figure E.6: Box stacking simulation task setup. Three different views are considered.

Box Stacking setup: The objective of this task is to plan a sequence of states leading the boxes to be stacked according to a goal image observation. Transitions between states, i.e., actions, can then be retrieved through the LSR [23]. The scene, shown in Figure E.6, is composed of four boxes that can be stacked in a 3×3 vertical grid. The boxes are interchangeable i.e., color does not matter, and each cell can only be occupied by one box at a time. The box stacking task has the following rules: *i*) a box can only be moved if there is no other box on top of it, *ii*) a box can only be placed on the ground or on top of another box (but never outside the grid), *iii*) no boxes can be added or completely removed from the grid (there are always 4 boxes in play). The arrangement of the objects in the scene represents the underlying state of the system.

Three different viewpoints are considered to capture the scene and four datasets are built as follows: *i*) \mathcal{BS}_f , where all the observations are taken from view front

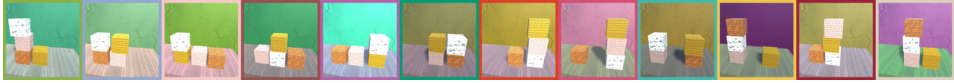


Figure E.7: Example observations of the 12 different possible states of the system. Note that the color of the boxes does not matter, i.e. boxes are interchangeable.

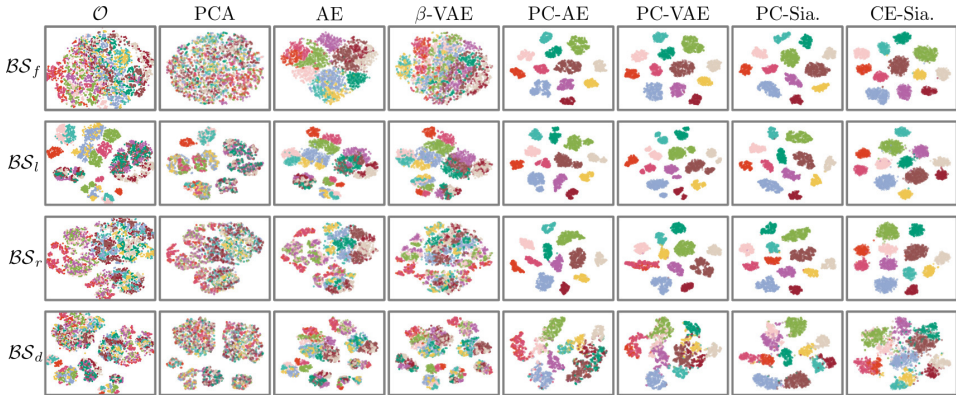


Figure E.8: Two-dimensional t-SNE plots of the latent representations from all models for the box stacking task. The rows show the results of the front view dataset (\mathcal{BS}_f), the left view (\mathcal{BS}_l), right view (\mathcal{BS}_r) and the different viewpoints dataset (\mathcal{BS}_d).

(blue in Figure E.6), *ii*) \mathcal{BS}_r , where the observations are only taken from view right (red in Figure E.6), *iii*) \mathcal{BS}_l , where the observations are only taken from view left (green in Figure E.6), and *iv*) \mathcal{BS}_d , where we enforce that the views for o_i and o_j in a training tuple are different. In each dataset, we randomly change lighting conditions, background, and table color in the observations. Moreover, we introduce a planar position noise of $\approx 17\%$ on the position of each box. Note that all of these changes are irrelevant factors of variation i.e., do not change the underlying system state for the box stacking task. For each dataset, we use 2500 data samples for training, with 1598 action pairs.

Underlying states: Given the box stacking rules, we can determine all the possible underlying states of the system. In particular, each state is given by a possible grid configuration specifying for every cell whether it is occupied by a box or not. Given the grid size and stacking rules, there are exactly 12 valid different box placements are shown in Figure E.7 and 24 legal state transitions.

Result and Discussion: We describe the results of this additional simulated box stacking task in the following section. In particular, we are interested to supplement the answer to the question from section 8: *Do contrastive-based losses outperform reconstruction-based losses when task-irrelevant factors of variations are present in the observations?*

Similar to the other two tasks, (box manipulation and shelf arrangement task), we carry out both a qualitative and quantitative analysis on the box stacking task. Regarding the qualitative analysis, we visualize the latent encodings obtained with the different models using the respective 2D t-SNE [38] plots in Figure E.8 for the box stacking task.

The top row shows the results from \mathcal{BS}_f , containing pairs taken from the frontal viewpoint. The second and third rows show the results with left \mathcal{BS}_l and right \mathcal{BS}_r views, respectively. Note that retrieving the system state from the observations in these datasets is more complex than in the frontal viewpoint dataset \mathcal{BS}_f since the boxes occupy different sized portions of the image depending on their location. Finally, the last row reports the results on the dataset \mathcal{BS}_d , where having observations taken from different viewpoints, making it extremely difficult to relate any visual changes to underlying state changes.

We can observe that raw observations as well as PCA fail completely in structuring the representations on all datasets. The purely reconstruction-based AE is able to exploit the visual similarity in the front view dataset (top row) to a certain extent structuring the encodings in a promising manner, even if not exactly separating different underlying states. However, this model clearly fails in the other datasets where few spurious clusters are formed in which the different states are mixed up. Similar results are obtained with the β -VAE, which shows a less structured latent space for the dataset \mathcal{BS}_f . A significant improvement of the latent space structure is recorded by introducing the PC loss term to the AE and β -VAE, leading to 12 visually distinct clusters for \mathcal{BS}_f , \mathcal{BS}_l and \mathcal{BS}_r (corresponding to the 12 underlying states of the system). The same applies to both purely contrastive loss-based Siamese networks (PC-Siamese and CE-Siamese). The different viewpoint dataset however is clearly more challenging for all models.

The qualitative observations based on t-SNE plots are confirmed by the numerical analysis reported in Table E.5. We observe that all models employing a contrastive loss (PC-AE, PC-VAE, PC-Siamese, and CE-Siamese) perform nearly perfectly for the frontal view dataset \mathcal{BS}_f (first row) in regard to all the performance metrics: the recorded homogeneity (h_c) and completeness (c_c) show a perfect clusterability and therefore enable the LSR to successfully plan in 100% of the cases (the exception being the CE-Siamese with 98.8% which indicates that only a start or goal state was misclassified). In contrast, directly using the raw observation, the PCA or reconstruction-based models achieve very low clustering and planning performance, reaching values $< 5\%$ for the existence of at least one correct path.

As far as the left viewpoint dataset is concerned (third row), all the models not employing any form of contrastive loss (PCA, AE, β -VAE) achieve low clustering (i.e., low h_c, c_c, s_c) and planning performance (i.e., low c_e , $\% any$ and $\% all$ with $\mathcal{V} \neq 12$ and $\mathcal{E} \neq 24$). The results show that the best performing model is the PC-AE that achieves 100% for $\% all$ and $\% any$ in the planning scores. It manages to build 16 clusters (only 4 clusters more than the ideal number) that have perfect homogeneity, completeness and a good silhouette score (0.85).

The other purely contrastive-based models (PC-Siamese and CE-Siamese) achieve

Models	Dataset \mathcal{BS}_f							
	\mathcal{V}	h_c	c_c	s_c	\mathcal{E}	c_e	Paths scores	
							% all	% any
-	118	0.61	0.62	0.29	119	0.78	2.53	2.8
PCA	107	0.67	0.72	0.33	123	0.79	4.04	4.20
AE	28	0.85	0.87	0.38	19	1.00	2.60	2.60
β -VAE	52	0.8	0.81	0.36	38	0.82	1.60	1.60
PC-AE	23	1.00	1.00	0.86	59	1.00	100.0	100.0
PC-VAE	19	1.00	1.00	0.84	48	1.00	100.0	100.0
PC-Sia.	18	1.00	1.00	0.84	44	1.00	100.0	100.0
CE-Sia.	13	1.00	1.00	0.35	28	0.96	98.80	98.80
Dataset \mathcal{BS}_d								
-	92	0.15	0.24	0.28	157	0.61	0.95	1.0
PCA	40	0.03	0.08	0.16	91	0.63	2.65	2.70
AE	21	0.04	0.12	0.13	43	0.58	1.07	1.20
β -VAE	29	0.05	0.10	0.05	49	0.61	2.17	2.40
PC-AE	80	0.92	0.92	0.26	136	0.89	59.61	63.20
PC-VAE	43	0.70	0.93	0.05	92	0.74	30.70	33.70
PC-Sia.	28	0.94	0.99	0.48	60	0.83	57.23	66.30
CE-Sia.	37	0.89	0.97	0.19	66	0.92	54.38	59.40
Dataset \mathcal{BS}_l								
-	131	0.34	0.53	0.35	192	0.68	5.4	5.6
PCA	85	0.32	0.51	0.27	154	0.75	13.74	15.90
AE	33	0.28	0.52	0.12	33	0.67	8.90	8.90
β -VAE	29	0.44	0.60	0.05	35	0.66	3.50	3.50
PC-AE	16	1.00	1.00	0.85	38	1.00	100.0	100.0
PC-VAE	18	0.90	1.00	0.75	40	1.00	76.30	76.30
PC-Sia.	17	0.91	1.00	0.72	39	0.87	56.67	64.20
CE-Sia.	16	0.92	1.00	0.33	39	0.87	61.40	69.20
Dataset \mathcal{BS}_r								
-	123	0.67	0.71	0.3	124	0.69	0.6	0.6
PCA	119	0.54	0.56	0.33	151	0.62	2.30	2.80
AE	35	0.71	0.74	0.33	24	0.88	1.10	1.10
β -VAE	41	0.76	0.79	0.37	32	0.81	1.20	1.20
PC-AE	21	1.00	1.00	0.74	54	0.96	90.70	95.40
PC-VAE	21	0.93	1.00	0.59	53	0.91	68.50	73.90
PC-Sia.	22	0.96	1.00	0.63	54	0.91	73.56	80.80
CE-Sia.	27	0.97	1.00	0.31	63	0.92	81.70	83.70

Table E.5: Evaluation results according to section 6 for the mapping models (rows 2 – 8 of each table) and the raw observation (first row of each table) on datasets \mathcal{BS}_f , \mathcal{BS}_d , \mathcal{BS}_l , and \mathcal{BS}_r from top to bottom for the simulated box stacking task. Best results in bold.

both lower performance of 65 – 70% and fall behind the PC-VAE with 76% planning performance for % any path. This is explained by imprecise clustering (shown by sub-optimal homogeneity scores ≈ 0.9) that compromises the planning performance.

Similar considerations also hold for results of the right viewpoint dataset in

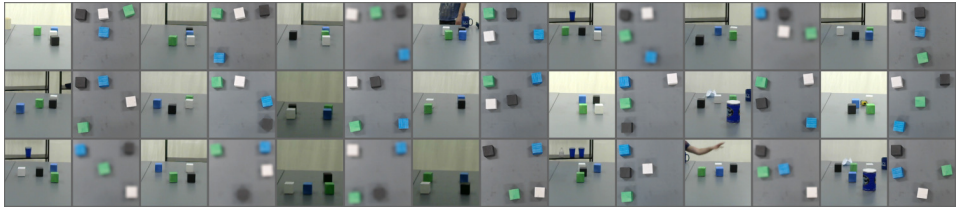


Figure E.9: 70/126 example observation from the side and top view for the box manipulation task.

Table E.5 (forth row). More specifically, poor clustering and planning metrics are achieved by the models that do not rely on the contrastive loss, while best general performance is obtained with PC-AE, which reaches 95.4% for % any as well as best clustering scores ($h_c = c_c = 1$ and $s_c = 0.74$). Acceptable planning performance is also achieved by the other models using the contrastive loss ($\approx 73 - 83\%$ for % any).

However, note that the good performance of the PC-AE does not translate to the dataset where different views are enforced (\mathcal{BS}_d) (second row). This suggests that it is able to combine reconstruction and contrastive losses beneficially as long as all the observations are obtained with a fixed viewpoint, but fails when the differences in observation become too large. In contrast, the purely contrastive models can effectively handle such differences. In this regard, it can be noticed that PC-AE, PC-Siamese, and CE-Siamese achieve the best (comparable) performance in terms of clustering and planning. Interestingly, the PC-AE is able to combine the reconstruction loss with the additional contrastive loss more effectively than the PC-VAE model. Possibly this is due to a better PC loss coefficient α choice or to the absence of the KL-term. Finally, on the performance of the other models on dataset \mathcal{BS}_d , we observe poor clustering and planning performance when using raw observations, PCA, AE, and β -VAE, confirming the relevance of task priors to handle task-irrelevant factors of variation.

In summary, we observe that models involving the contrastive term (PC-AE, PC-VAE, PC-Siamese, CE-Siamese) outperform the ones that do not (PCA, AE, β -VAE) by a significant margin.

10.2 Box Manipulation Setup and Underlying States

Box Manipulation setup: The box manipulation task involves 4 boxes which are arranged in a predetermined 3×3 grid. The rules for the task are the following: *i)* a box can not be moved into a cell that already contains a box, *ii)* a box can only be moved into the four cardinal directions (as it happens in a box pushing task), *iii)* a box can only be moved one cell at the time. A similar pair is done by swapping the position of boxes and an action pair by performing a random move. Before executing an action we check its validity (preconditions) as described in 7.

During the collection, a number of task-irrelevant objects (hat, coffee-mug, mask, etc.) are placed in the view field of the side camera and people are moving in the background along with several other objects. The top view often performs auto focus when the robotic arm is moving the boxes which lead to blurry observations. Example observations are shown in Figure E.9.

In total 2135 training data pairs were collected in a self-supervised manner in ≈ 30 hours. As the action between states are reversible, we also included the reversed version of each action pair, i.e., given the tuple $(o_1, o_2, s = 0)$ we add $(o_2, o_1, s = 0)$.

Underlying states: Given the box manipulation rules, we can determine all the possible underlying states of the system. In particular, each state is given by a possible grid configuration specifying for every cell whether it is occupied by a box or not. Given the grid size and manipulation rules, there are exactly 126 different box placements, i.e. grid configurations.

10.3 Shelf Arrangement Setup and Underlying States

Shelf arrangement setup: The shelf arrangement setup is composed of a table and two shelves. The table has four potential slots for task relevant objects. For the shelves, each shelf slot can be occupied at most by one task relevant object. The task object can however be placed either to the left or the right inside the shelf itself. The task relevant objects are always present in the scene. An action moves a relevant object from the table to the shelf or vice-versa while a swapping motion (a similar pair) exchanges the position of two objects. A small amount of positional noise for the objects is also introduced each time the scene is generated. Five distractor objects, that are not relevant for the task, can be present in the shelf slots. Their position inside the shelf slots is not fixed.

Underlying states: The rules for the shelf arrangement task result in a system that has exactly 70 underlying distinct states and 320 legal transitions, which are obtained by considering all possible combinations of object configurations in the 8 available slots. Figure E.10 shows examples of all 70 distinct underlying states for the dataset with no distractors present.

10.4 Architectures and Hyperparameters

In this section, we describe the architectural details, as well as the hyperparameters for all considered models. The input dimension for all models is a $256 \times 256 \times 3$ image. To make the comparison between the models as fair as possible, each uses the same training data and latent space dimension.

PCA: We used the popular scikit-learn [39] implementation of the principal component analysis based on [40]. The number of components was set to 12 and the model fit to the training dataset. The dimension reduction was then applied to the holdout dataset for evaluation.

AE: The implementation of encoder and decoder for the Auto-encoder [28] relies on the ResNet architecture in [41], with a depth of two per block for the box stacking



Figure E.10: Example observations of the 70 different possible states of the self arrangement system. Note that the individual object does not matter, i.e. objects are interchangeable.

and manipulation tasks and a larger ResNet architecture having six layers with depth two for the shelf stacking task. We train each model for 500 epochs and a batch size of 64.

β -VAE: The implementation of encoder and decoder of the β -VAE [29] is realized by adding the probabilistic components to the AE architecture. We train the models for 500 epochs with a scheduling for beta from 0 to 1.5 and a batch size of 64.

PC-AE: The PC-AE uses the same architecture as the AE model.

PC-VAE: The PC-VAE uses the same architecture as the β -VAE model.

PC-Siamese:

The Siamese network [42] architecture is comprised of two identical encoder networks. Each encoder has the following latent encoding architecture:

$$\begin{aligned} x_1 &= \text{MaxPool}(x, 2 \times 2) \\ x_2 &= \text{Conv}(x_1, 4 \times 4, \text{relu}) \\ x_3 &= \text{Conv}(x_2, 4 \times 4, \text{relu}) \\ x_4 &= \text{MaxPool}(x_3, 7 \times 7) \\ z &= \text{FC}(x_4, 12, \text{relu}) \end{aligned}$$

CE-Siamese: This model has the same architecture as the Siamese but uses the normalized temperature-scaled cross entropy loss.

Hyperparameters: The latent space dimension was set to 12 for all 56 models. Concerning the hyperparameters in the loss functions, we employed the same scheduling as in [23] for α and γ for the losses employed in PC-AE and PC-VAE

and for β in (4), reaching $\alpha = 100$, $\gamma = 2500$ and $\beta = 2$. In order to set the minimum distance d_m in (E.1) for PC-AE and PC-VAE, we leveraged the approach in [23] based on measuring the average distance of the action pairs in the models AE and β -VAE, while we set it to 0.5 for PC-Siamese. The AE, β -VAE, PC-AE and PC-VAE were trained for 500 epochs while the PC-Sia. and CE-Sia were trained for 100 epochs. Concerning the HDBSCAN parameter, denoting the minimum number of samples in each cluster, we set it to 5 for the datasets obtained in simulation and to 2 for the real-world datasets. This is motivated by the fact that a smaller amount of data is available for the real-world setting.

10.5 Additional Experimental Results for Box Manipulation

Here we mention some additional results for the box manipulation task.

Experimental Results: Table E.6 and Table E.7 show all the results for the box manipulation task for all 7 mapping models, as well as for the raw observations, on the top view (\mathcal{BM}_t - first row in the table), side (\mathcal{BM}_s - third row in the table) and mixed view (\mathcal{BM}_{st} - Table E.7) datasets as well as their augmented versions ($\overline{\mathcal{BM}}_t$, $\overline{\mathcal{BM}}_s$, $\overline{\mathcal{BM}}_{st}$) below them respectively. We will focus the discussion on the results that were omitted in the discussion in section 8, i.e., we will focus on the side view dataset.

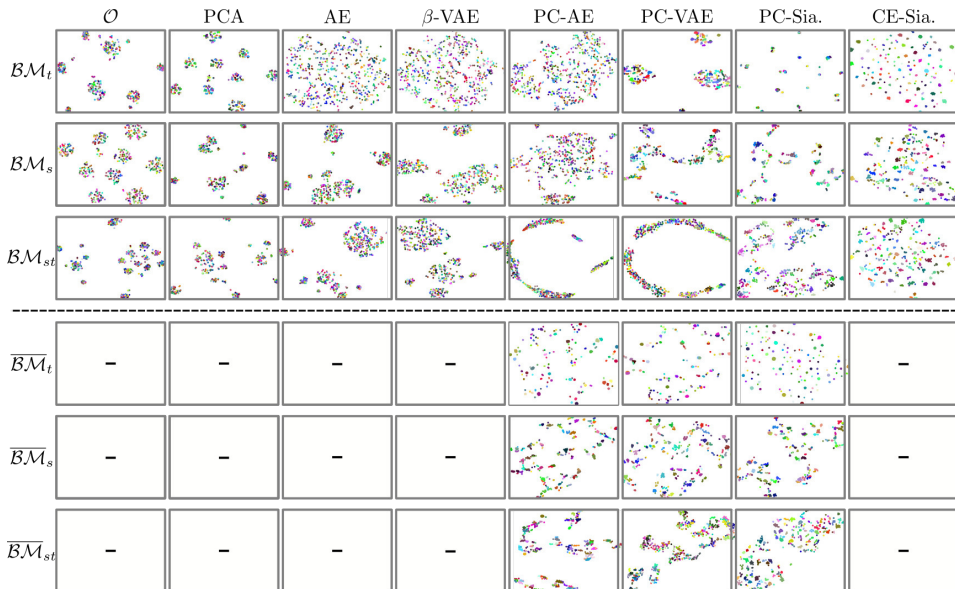


Figure E.11: Two-dimensional t-SNE plots of the latent representations from all models for the box manipulation task. The rows shows the results of the top view (\mathcal{BM}_t), the side view (\mathcal{BM}_s), and the mixed viewpoints dataset (\mathcal{BM}_{st}), as well as their augmented versions in descending order.

Concerning the side view dataset, we observe that, as for \mathcal{BM}_t and \mathcal{BM}_{st} , the CE-Siamese achieves overall best planning and clustering performance, reaching 72.2% for $\% any$ and almost perfect homogeneity and completeness ($h_c = c_c = 0.99$) with 298 clusters. General lower performance is obtained by the other pure contrastive-based model PC-Siamese, which reaches 34.2% for $\% any$ and creates a higher number of clusters ($|\mathcal{V}| = 345$) which are less homogeneous and complete ($h_c = 0.5, c_c = 0.98$). A further decrease on the planning performance is observed with PC-VAE (10.0% for $\% any$) which leads to creating a large number of clusters ($|\mathcal{V}| = 548$) with good homogeneity and completeness ($h_c = c_c = 0.99$). All the other models (PCA, AE, β -VAE, and PC-AE) are getting 0% planning performance.

Regarding the augmentation, we do not evaluate the case of the raw observations or the models not using any contrastive component (PCA, AE, β -VAE) since such models do not exploit the random action pairs generated by the augmentation. Similarly, we do not evaluate CE-Siamese as it does not use the action pairs in the dataset. Concerning the other models, interestingly the augmentation propels the PC-AE from having 0% for $\% any$ on the dataset \mathcal{BM}_s to having 44.9% for the augmented version $\overline{\mathcal{BM}}_s$. An improvement of $\approx 25\%$ is also recorded for the PC-VAE and PC-Siamese, with the latter outperforming the former (31.5% for PC-VAE, 55.6% for PC-Siamese for the score $\% any$).

Models	Dataset \mathcal{BM}_t							
	\mathcal{V}	h_c	c_c	s_c	\mathcal{E}	c_e	Paths scores	
							% all	% any
-	1016	0.92	0.92	0.79	583	0.78	0.0	0.0
PCA	496.0	0.75	0.78	0.52	452.0	0.52	0.0	0.0
AE	233.0	0.49	0.57	0.29	234.0	0.27	0.0	0.0
β -VAE	539.0	0.85	0.85	0.51	422.0	0.62	0.0	0.0
PC-AE	246.0	0.54	0.6	0.3	258.0	0.28	0.0	0.0
PC-VAE	570.0	1.0	1.0	0.58	488.0	1.0	29.9	29.9
PC-Sia.	389.0	0.99	1.0	0.52	458.0	0.97	47.37	57.3
CE-Sia.	150.0	1.0	1.0	0.67	325.0	1.0	98.3	98.3
	Dataset \mathcal{BM}_t							
PCA	-	-	-	-	-	-	-	-
AE	-	-	-	-	-	-	-	-
β -VAE	-	-	-	-	-	-	-	-
PC-AE	218.0	0.99	1.0	0.71	375.0	0.98	72.12	82.5
PC-VAE	395.0	1.0	1.0	0.56	461.0	1.0	89.7	89.7
PC-Sia.	133.0	1.0	1.0	0.9	314.0	1.0	97.7	98.2
CE-Sia.	-	-	-	-	-	-	-	-
	Dataset \mathcal{BM}_s							
-	1002	0.96	0.96	0.78	558	0.89	0.0	0.0
PCA	539.0	0.74	0.75	0.59	479.0	0.35	0.0	0.0
AE	532.0	0.77	0.78	0.57	459.0	0.41	0.0	0.0
β -VAE	531.0	0.79	0.8	0.59	456.0	0.45	0.0	0.0
PC-AE	550.0	0.79	0.8	0.59	456.0	0.47	0.0	0.0
PC-VAE	548.0	0.99	0.99	0.57	460.0	0.98	10.0	10.0
PC-Sia.	345.0	0.95	0.98	0.48	391.0	0.91	31.02	34.2
CE-Sia.	298.0	0.99	0.99	0.41	407.0	0.94	71.57	72.2
	Dataset \mathcal{BM}_s							
PCA	-	-	-	-	-	-	-	-
AE	-	-	-	-	-	-	-	-
β -VAE	-	-	-	-	-	-	-	-
PC-AE	420.0	0.98	0.99	0.53	456.0	0.95	40.89	44.9
PC-VAE	519.0	1.0	1.0	0.57	448.0	0.99	31.5	31.5
PC-Sia.	304.0	0.97	0.99	0.5	394.0	0.92	46.6	55.6
CE-Sia.	-	-	-	-	-	-	-	-

Table E.6: Evaluation results of the models for the box manipulation task. Datasets \mathcal{BM}_t (first row), and \mathcal{BM}_s (third row) with respective augmented versions reported below them. Best results in bold.

The quantitative analysis is also reflected in the t-SNE plots in Figure E.11. We can observe that only CE-Siamese obtains a good separation for all datasets. Moreover, the random sampling data augmentation significantly improves the structure of the latent spaces obtained by PC-AE, PC-VAE and PC-Siamese for the top and side views, while lower improvement is recorded for the mixed view dataset.

10.6 Additional Experimental Results for Shelf Arrangement

Table E.8 reports the evaluation results with all the datasets in the shelf arrangement task. The first and second rows reports the datasets that have no distractor

Models	Dataset \mathcal{BM}_{st}							
	$ \mathcal{V} $	h_c	c_c	s_c	$ \mathcal{E} $	c_e	Paths scores	
							% all	% any
-	710	0.83	0.83	0.5	496	0.58	0.0	0.0
PCA	400.0	0.59	0.62	0.46	453.0	0.25	0.0	0.0
AE	554.0	0.87	0.88	0.56	454.0	0.69	0.0	0.0
β -VAE	407.0	0.72	0.74	0.44	361.0	0.38	0.0	0.0
PC-AE	318.0	0.62	0.91	0.61	325.0	0.47	0.0	0.0
PC-VAE	381.0	0.84	0.85	0.42	295.0	0.65	0.1	0.1
PC-Sia.	289.0	0.96	0.96	0.4	312.0	0.92	26.34	27.5
CE-Sia.	232.0	0.99	0.99	0.41	354.0	0.99	78.39	78.7
Dataset \mathcal{BM}_{st}								
PCA	-	-	-	-	-	-	-	-
AE	-	-	-	-	-	-	-	-
β -VAE	-	-	-	-	-	-	-	-
PC-AE	383.0	0.96	0.98	0.53	433.0	0.88	40.77	48.9
PC-VAE	335.0	0.91	0.92	0.42	274.0	0.84	0.7	0.7
PC-Sia.	235.0	0.99	0.99	0.41	337.0	0.99	77.7	78.8
CE-Sia.	-	-	-	-	-	-	-	-

Table E.7: Evaluation results of the models for the box manipulation task. Dataset \mathcal{BM}_{st} (first row) with respective augmented versions reported below. Best results in bold.

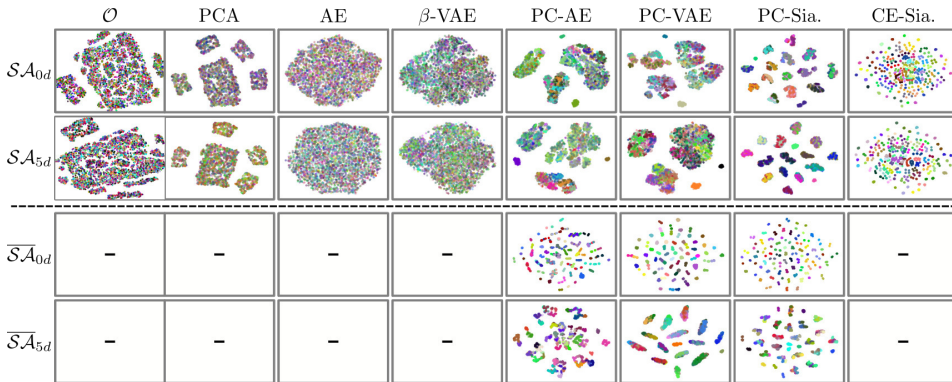


Figure E.12: Two-dimensional t-SNE plots of the latent representations from all models for the shelf arrangement task. All the data augmentations are considered.

present, while the third and fourth shows the performance for the dataset where all five distractors are present (with probability 0.8). The first and third row are all models for the non-augmented datasets (\mathcal{SA}_{0d} and \mathcal{SA}_{5d}). The second and fourth row block indicates the datasets augmented with randomly sampled dissimilar pairs like described in section 3 ($\overline{\mathcal{SA}}_{0d}$ and $\overline{\mathcal{SA}}_{5d}$). Note that the CE-Siamese is not shown for the augmentations that only alter the dissimilar pairs as they are not used in this particular model.

Concerning the non-augmented datasets \mathcal{SA}_{0d} and \mathcal{SA}_{5d} , also the performance

Models	Dataset \mathcal{SA}_{0d}							Paths scores	
	$ \mathcal{V} $	h_c	c_c	s_c	$ \mathcal{E} $	c_e			
							$\% all$	$\% any$	
-	2	0.0	0.33	0.14	2	0.0	0.0	0.0	
PCA	3	0.05	0.29	0.17	5	0.00	0.00	0.00	
AE	2	0.00	0.38	0.17	2	0.00	0.00	0.00	
β -VAE	13	0.36	0.56	0.07	11	0.18	0.00	0.00	
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	1.80	
CE-Sia.	296	1.00	1.00	0.54	842	1.00	95.9	95.90	
Dataset $\overline{\mathcal{SA}}_{0d}$									
PCA	-	-	-	-	-	-	-	-	
AE	-	-	-	-	-	-	-	-	
β -VAE	-	-	-	-	-	-	-	-	
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	
CE-Sia.	-	-	-	-	-	-	-	-	
Dataset \mathcal{SA}_{5d}									
-	1	0.0	0.36	0.06	0	0.0	0.0	0.0	
PCA	5	0.09	0.30	0.18	9	0.22	0.00	0.00	
AE	2	0.01	0.43	0.00	2	0.00	0.00	0.00	
β -VAE	2	0.01	0.52	0.08	2	0.00	0.00	0.00	
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 $\overline{\mathcal{SA}}_{5d}$									
PCA	-	-	-	-	-	-	-	-	
AE	-	-	-	-	-	-	-	-	
β -VAE	-	-	-	-	-	-	-	-	
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	
CE-Sia.	-	-	-	-	-	-	-	-	

Table E.8: Evaluation results of the models for the shelf arrangement task. Datasets \mathcal{SA}_{0d} (first row) and \mathcal{SA}_{5d} (third row) with respective augmented versions below them are considered. Best results in bold.

of models PCA, AE and β -VAE are reported in addition to the ones in the section 8 confirming that none of the non-augmented datasets (except for CE-Siamese) achieve any meaningful performance. Regarding the augmented datasets, we can observe they lead to much better structured latent spaces for all models in the case of no distractors. Especially the PC-Siamese model achieves perfect clustering score for the homogeneity and compactness, as well as perfect planning performance of 100% for both $\% any$ and $\% all$. However, the augmentation only helps to a marginal extent for the case of five distractor objects present, where only the

PC-Siamese model improves to 13.1% for % any.

The numbers are easily confirmed with visually inspecting the t-SNE plots in Figure E.12. We can see that a good separation is only achieved throughout all datasets from the CE-Siamese models. The random sampling data augmentation in section 3 helps the models for the no distractor datasets.

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,” in *Int. Conference on Robotics and Automation (ICRA) - Workshop on Autonomous Learning* 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," in *International Conference on Learning Representations (ICLR)*, 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. Kavvaki, 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.
39. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
40. M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.
41. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
42. D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2020.

