# Latent Space Roadmap for Visual Action Planning

Martina Lippi*[1,2], Petra Poklukar*[1], Michael C. Welle*[1], Anastasiia Varava[1],
Hang Yin[1], Alessandro Marino[3], and Danica Kragic[1]

***Abstract*—We present a framework for visual action planning of complex manipulation tasks with high-dimensional state spaces such as manipulation of deformable objects. Planning is performed in a low-dimensional latent state space that embeds images. We define and implement a Latent Space Roadmap (LSR) which is a graph-based structure that globally captures the latent system dynamics. Our framework consists of two main components: a Visual Foresight Module (VFM) that generates a visual plan as a sequence of images, and an Action Proposal Network (APN) that predicts the actions between them. We show the effectiveness of the method on a simulated box stacking task as well as a T-shirt folding task performed with a real robot.**

## I. Introduction and Related Work

Designing efficient state representations for task and motion planning is a fundamental problem in robotics studied for several decades [1], [2]. Traditional planning approaches rely on a comprehensive knowledge of the state of the robot and the surrounding environment as for example [3] and [4].

However, when dealing with high-dimensional state spaces and complex dynamics, such as highly deformable objects, these approaches become intractable [5] even when sampling-based algorithms [6] are deployed. For this reason, data-driven low-dimensional latent space representations for planning are receiving increasing attention as they make it possible to consider states that would otherwise be intractable. In particular, deep neural networks allow for implicit representation of complex state spaces and their dynamics, thus enabling an automatic extraction of lower-dimensional state representations [7]. Some of the most common approaches to learning compact representations in an unsupervised fashion are latent variable models such as Variational Autoencoders (VAEs) [8], [9] or encoder-decoder based Generative Adversarial Networks (GANs) [10], [11]. These models can learn low-dimensional state representations directly from images. In this way, images can be used as input for planning algorithms to generate *"visual plans"* [12].

Latent state representations, however, are not guaranteed to capture the *global structure* and dynamics of the system, *i.e.* to encode all the possible system states and respective feasible transitions. Furthermore, not all points in the latent space necessarily correspond to physically *valid* states of the system, which makes it hard to plan by naively interpolating between start and goal states.

One way to address these shortcomings is to restrict the exploration of the latent space via imitation learning as presented in [13], where a latent space Universal Planning Network (UPN) embeds differentiable planning policies and the process is learned in an end-to-end fashion. A more common solution to mitigate the challenges of planning in latent spaces is to collect a large amount of training data that densely covers the state space and allows to infer dynamically valid transitions between states. Following this approach, the authors in [12] propose a framework for *global search* in the latent space where a network approximates the latent space dynamics and motion planning is performed by an RRT-based algorithm. Similarly, the manipulation of a deformable rope to desired goal states is investigated in [14] where 500 hours worth of data collection are used to learn the rope's inverse dynamics.

In this paper, we address the aforementioned challenges related to latent space representations by constructing a Latent Space Roadmap (LSR) which is a graph-based structure built in the latent space to both capture the *global structure* of the state space and avoid sampling *invalid* states. In this way, our method, visualised in Fig. 1, is able to identify the feasible transitions between regions containing similar states using the LSR and to generate a valid *visual action plan* by sampling new valid states inside these regions. Furthermore, our approach is data-efficient as we do not assume that the training dataset densely covers the state space neither accurately represents system dynamics. We instead consider a dataset consisting of *pairs* of images and demonstrated actions connecting them, and then learn feasible transitions between states from this *partial* data. This allows avoiding full imitation for modeling as in the UPN framework [13] as well as tackling tasks involving highly-deformable objects such as cloths. Finally, we experimentally evaluate our method on a

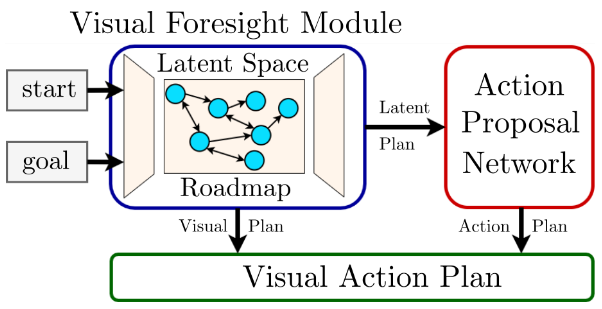

Fig. 1: Overview of the proposed method. The Visual Foresight Module (blue) takes the start and goal images and produces a visual plan from a latent plan found with the Latent Space Roadmap (cyan). The Action Proposal Network (red) proposes suitable actions to achieve the transitions between states in the visual plan. The final result is a *visual action plan* (green) from start to goal containing actions to transition between consecutive states.

*These authors contributed equally (listed in alphabetical order).
[1]KTH Royal Institute of Technology Stockholm, Sweden
[2]University of Salerno, Salerno, Italy
[3]University of Cassino and Southern Lazio, Cassino, Italy

simulated box stacking task as well as a real-world T-shirt folding task.

## II. Problem Statement and Notation

The goal of visual action planning 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. The problem is formalized in the following.

Let $\mathcal{I}$ be the state space of the system represented as images with fixed resolution and let $\mathcal{I}_{sys} \subset \mathcal{I}$ be the subset of *valid states* representing all the states of the system that are possible to reach while performing the task. Let $\mathcal{U}$ be the set of possible control inputs or actions.

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

To reduce the complexity of the problem we consider a lower-dimensional latent space $\mathcal{Z}$ encoding $\mathcal{I}$, and $\mathcal{Z}_{sys} \subset \mathcal{Z}$ encoding $\mathcal{I}_{sys}$. Using $\mathcal{Z}_{sys}$, a visual plan can be computed in the latent space as $P_z = \{z_{start} = z_0, z_1, ..., z_N = z_{goal}\}$ where $z_n \in \mathcal{Z}_{sys}$, and then decoded as a sequence of images. In order to obtain a valid visual plan, we study the structure of the space $\mathcal{Z}_{sys}$ which in general is not path-connected. To ensure a valid $P_z$, we make an $\varepsilon$-validity assumption that is motivated by the continuity of the encoding of $\mathcal{I}$ into $\mathcal{Z}$:

*Assumption 1:* Let $z \in \mathcal{Z}_{sys}$ be a valid latent state. Then there exists $\varepsilon > 0$ such that any other latent state $z'$ in the $\varepsilon-$neighborhood $N_\varepsilon(z)$ of $z$ is a valid latent state, *i.e.* the following equivalence relation holds true

$$z \sim z' \iff ||z - z'||_d < \varepsilon, \tag{1}$$

where the subscript $d \in \{1, 2, \infty\}$ denotes the metrics $L_1, L_2$ and $L_\infty$, respectively, and $\varepsilon$ a task-dependent parameter.

Therefore, given a set of valid latent states $z_i \in \mathcal{Z}_{sys}$, with $i = 1, ..., M$, the union $\mathcal{R}_z^\varepsilon$ of their $\varepsilon-$neighborhoods consists of valid points:

$$\mathcal{R}_z^\varepsilon = \bigcup_{i=1}^{M} N_\varepsilon(z_i) \subset \mathcal{Z}_{sys}. \tag{2}$$

The subspace $\mathcal{R}_z^\varepsilon$ can be then considered as the union of $m$ path-connected components called *valid regions* and denoted by $\{\mathcal{Z}_{sys}^i\}_{i=1}^m$. To connect them, we define a set of transitions:

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

Given a set of valid regions $\{\mathcal{Z}_{sys}^i\}_{i=1}^m$ 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. 2. To this end, we define a Latent Space Roadmap:

*Definition 3:* A Latent Space Roadmap is a directed graph $LSR = (\mathcal{V}_{LSR}, \mathcal{E}_{LSR})$ where each vertex $v_i \in \mathcal{V}_{LSR} \subset \mathcal{Z}_{sys}$ for $i \in \{1, 2, ..., m\}$ is an equivalence class representative of the valid region $\mathcal{Z}_{sys}^i \subset \mathcal{Z}_{sys}$, and an edge $e_{i,j} = (v_i, v_j) \in \mathcal{E}_{LSR}$ represents a transition function $f_z^{i,j}$ between the corresponding valid regions $\mathcal{Z}_{sys}^i$ and $\mathcal{Z}_{sys}^j$ for $i \neq j$.

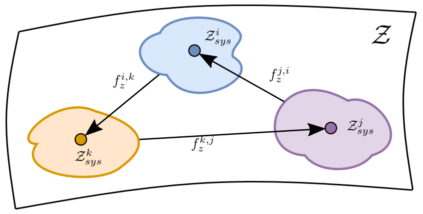

Fig. 2: A visualisation of the structure of the latent state space with valid regions $Z_{sys}^i$ and transition functions $f_z^{j,i}$ between them.

## III. An overview of our approach

We consider a training dataset $\mathcal{T}_I$ consisting of generic tuples of the form $(I_1, I_2, \rho)$ where $I_1 \subset \mathcal{I}_{sys}$ is an image of the start state, $I_2 \subset \mathcal{I}_{sys}$ an image of the successor state, and $\rho$ a variable representing the action that took place between the two states. Here, an action is considered to be a *single* transformation that produces any consecutive state $I_2$ different from the start state $I_1$, *i.e.*, $\rho$ cannot be a composition of several transformations. On the contrary, no action was performed if states $I_1$ and $I_2$ are variations of the same state. The variable $\rho = (a, u)$ consists of a binary variable $a \in \{0, 1\}$ for action\no action, and $u$ containing the task-dependent action-specific information which can be used to infer the transition functions $f_z^{i,j}$.

Our method consists of two main components depicted in Fig. 1. The first is the Visual Foresight Module (VFM) which is a trained VAE endowed with a Latent Space Roadmap (LSR). Given a start and goal state, the VFM produces a visual plan $P_I$ consisting of a sequence of images. The sequence $P_I$ is a decoded latent plan $P_z$ found in the VAE's latent space using the LSR. The second component is the Action Proposal Network (APN) which takes a pair $(z_i, z_{i+1})$ of consecutive latent states from the latent plan $P_z$ produced by the VFM and proposes an action $u_i$ to achieve the desired transition $f_z^{i,i+1}(z_i, u_i) = z_{i+1}$. The two components combined produce a visual action plan that can be executed by any suitable framework.

## IV. Visual Foresight Module (VFM)

The Visual Foresight Module in Fig. 1 has two building blocks that are trained in a sequential manner. Firstly, we train a VAE with an additional term in the loss function that affects the structure of the latent space. Once the VAE is trained, we build our LSR in its latent space $\mathcal{Z}$ which identifies the valid regions $\mathcal{Z}_{sys}^i$. We present the details below.

*1) Latent state space:* Let $I \subset \mathcal{I}_{sys}$ be an input image, and let $z$ denote the unobserved latent variable and $p(z)$ the prior distribution. The VAE model [8], [9] is trained to minimize

$$\mathcal{L}_{vae}(I) = E_{z\sim q(z|I)}[\log p(I|z)] + \beta \cdot D_{KL}\left(q(z|I)||p(z)\right) \tag{3}$$

with respect to the parameters of the encoder and decoder neural networks which model the parameters of the approximate posterior distribution $q(z|I)$ and the likelihood function

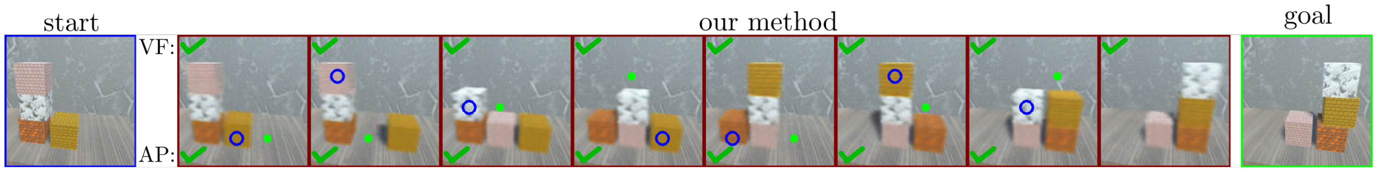

Fig. 3: An example of a visual action plan from the start (left) to the goal state (right) for the box stacking task produced using our method.

$p(I|z)$, respectively. Since our training data consists of tuples $(I_1, I_2, a)$, we compute $\mathcal{L}_{vae}$ for $I_1$ and $I_2$ separately and leverage the information contained in the binary variable $a$ by minimizing an additional *action* term

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

where $z_1, z_2 \subset \mathcal{Z}_{sys}$ are the latent encodings of the input states $I_1, I_2 \subset \mathcal{I}_{sys}$, respectively, and the subscript $d$ denotes the metric as in (1). The hyperparameter $d_m$ introduced among the action pairs enforces different states to be encoded in separate parts of the latent space. The action term $\mathcal{L}_{action}$ naturally encourages the formulation of the valid regions $\mathcal{Z}^i_{sys}$ in the latent space while maintaining the capability to generalise, *i.e.* to sample novel valid states inside each region $\mathcal{Z}^i_{sys}$.

The complete VAE loss term then equals

$$\mathcal{L}(I_1, I_2) = \frac{1}{2}(\mathcal{L}_{vae}(I_1) + \mathcal{L}_{vae}(I_2)) + \gamma \cdot \mathcal{L}_{action}(I_1, I_2) \quad (5)$$

where the parameter $\gamma$ controls the influence of the distances among the latent codes on the structure of the latent space.

### A. Latent Space Roadmap (LSR)

The Latent Space Roadmap is defined in *Definition 3* and based on the idea that each node in the roadmap is associated with a valid region $\mathcal{Z}^i_{sys}$. Two nodes are connected by an edge if there exists an action pair $(I_1, I_2, \rho)$ in the training dataset $\mathcal{T}_I$ such that the transition $f_z^{1,2}(z_1, u_1) = z_2$ is achieved in $\mathcal{Z}_{sys}$.

We provide the summary of our algorithm but refer the reader to [15] for full details. In Phase 1, we build a *reference* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ induced by $\mathcal{T}_z$ which serves as a look-up graph to keep track of which areas in $\mathcal{Z}_{sys}$ have already been explored as well as to preserve the edges that later induce the transition functions $f_z^{i,j}$. In Phase 2, we identify the valid regions $\mathcal{Z}^i_{sys} \subset \mathcal{Z}_{sys}$ using DBSCAN [16] with distance parameter $\varepsilon$. In Phase 3, we build the LSR $= (\mathcal{V}_{LSR}, \mathcal{E}_{LSR})$ such that each node $v_i \in \mathcal{V}_{LSR}$ is a representative of the valid region $\mathcal{Z}^i_{sys}$ and the edges between them are inferred using $\mathcal{E}$ from the reference graph. 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 valid regions.

The parameter $\varepsilon$ is obtained as a weighted sum of the mean and standard deviation of the distances $\|z_1 - z_2\|_d$ among the no-action latent pairs such that similar states, captured in the no-action pairs, belong to the same valid region, while states in the action pairs are allocated to different valid regions.

Using the LSR and the trained VAE model, we can generate one or more visual action plans from start to goal state. To this aim, the states are first encoded in the latent space and the closest nodes in the LSR are found. Next, all shortest paths [17] in the LSR between the identified nodes are retrieved.

## V. Action Proposal Network (APN)

The Action Proposal Network is used to predict the specifics of an action $u_i$ that occurs between a latent pair $(z_i, z_{i+1})$ from a latent plan $P_z$ produced by the VFM. We deploy a diamond-shaped multi layer perceptron and train it in a supervised fashion on the latent *action* pairs $(z_1, z_2, \rho = (1, u))$ obtained from the enlarged dataset $\mathcal{T}_z$ by leveraging sampling in the latent space.

## VI. Experiments

The proposed approach is evaluated on a simulated box stacking task and a T-shirt folding task on a real robot.

### A. Box stacking

The simulation setup, shown in Fig. 3 is composed of four boxes with different textures that can be stacked in a $3 \times 3$ grid. The action-specific information $u$ is a pair $u = (p, r)$ of pick $p$ and release $r$ coordinates in the grid.

For the VFM, we deploy a baseline VAE-b with $\gamma = 0$, and three distance VAEs (VAE-$L_1$, VAE-$L_2$, VAE-$L_\infty$) with a ResNet architecture [18] for the encoder and decoder networks and a 64-dimensional latent space. Similarly, we train four APNs (APN-b, APN-$L_1$, APN-$L_2$ and APN-$L_\infty$). The designed task contains exactly 288 different grid configurations. Given a pair of such grid configurations and the ground truth stacking rules, it is possible to analytically determine whether or not an action is allowed between them. This enables an automatic evaluation of the structure of the latent space $\mathcal{Z}_{sys}$, the quality of the visual plan $P_I$ generated by the VFM as well as of the corresponding action plan $P_u$ predicted by the APN.

*1) VAE latent space analysis:* Let each of the 288 possible grid configuration represent a class. We analyse the difference between the minimum inter-class distance and the maximum intra-class distance for each class. The higher the value the better separation of classes in the latent space is achieved. When the latent states are obtained using VAE-b we observe the difference to be always negative with an average value of $\approx -8.3$. On the other hand, when calculated on points encoded with VAE-$L_1$ it becomes non-negative for 286/288 classes and its mean value increases to $\approx 0.78$. This means that, even when there exists no direct link between two samples of different classes and thus the action term for the pair is never activated, the VAE-$L_1$ is able to encode them such that the desired distances in the latent space are respected. We therefore conclude that the action term results in a better structured latent space $\mathcal{Z}_{sys}$.

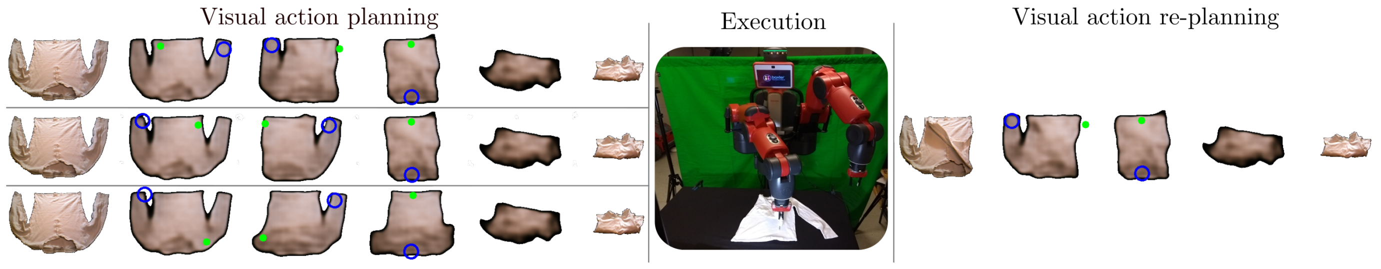

Fig. 4: Execution of the folding task with re-planning. On the left, a set of initial visual action plans reaching the goal state is proposed. After the first execution, only one viable visual action plan remains.

| Model | All | Any | Trans. |
|---|---|---|---|
| VAE-b + LSR-$d$, $\forall d$ | 0 % | 0 % | 33.3 % |
| VAE-$L_1$ + LSR-$L_1$ | **100 %** | **100 %** | **100 %** |
| VAE-$L_2$ + LSR-$L_2$ | 99.9 % | 99.9 % | 99.9 % |
| VAE-$L_\infty$ + LSR-$L_\infty$ | 8.2 % | 12 % | 53.2 % |

Table I: Visual foresight results for box stacking case study comparing different metrics (best results in bold).

*2) LSR and APN analysis:* In Table I we show the results obtained on LSRs built with the training data from the baseline VAE (first row) and the action VAEs (last three rows). In particular, we report the percentage of cases when all the shortest paths in each LSR are correct, when at least one of the proposed paths is correct, and the percentage of correct single transitions. All APNs perform with 99% or higher accuracy evaluated on 10 different random seedson an unseen test set consisting of 1491 action pairs.

## B. T-shirt folding

A Baxter robot, equipped with a Primesense RGB-D camera mounted on its torso, is used to fold a T-shirt in different ways as shown in Fig. 4. We perform a re-planning step after each action execution to account for possible uncertainties. The current cloth state is then considered as a new start state and a new visual action plan is produced until the goal state $I_{goal}$ is reached or the task is terminated. Compared to the box stacking task we use a larger version of the ResNet architecture for the VFM but keep the 64-dimensional latent space.

*1) APN Analysis:* We evaluate the performance of the APN models on 5 random seeds on a test split consisting of 104 action pairs. Table II reports mean and standard deviation of the Mean Squared Error calculated across the different random seeds. We observe a higher error when using VAE-b which again indicates that the latent space lacks structure if the action term (4) is excluded from the loss function.

| Model | Pick | Release | Total |
|---|---|---|---|
| APN-b | $0.50 \pm 0.09$ | $0.68 \pm 0.09$ | $1.24 \pm 0.16$ |
| APN-$L_1$ | $\mathbf{0.34 \pm 0.06}$ | $\mathbf{0.49 \pm 0.09}$ | $\mathbf{0.87 \pm 0.07}$ |
| APN-$L_2$ | $0.40 \pm 0.05$ | $0.51 \pm 0.08$ | $0.96 \pm 0.12$ |
| APN-$L_\infty$ | $0.49 \pm 0.05$ | $0.65 \pm 0.07$ | $1.20 \pm 0.07$ |

Table II: The error of action predictions obtained in the folding task on APN models with different metrics (best results in bold).

*2) Execution Results:* The performance of the entire system cannot be evaluated in an automatic manner as in the box stacking task. We therefore choose five novel goal configurations and perform the folding task five times per configuration on each framework F-$L_d$ that uses VAE-$L_d$, APN-$L_d$, and LSR-$L_d$ with $d = 1, 2, \infty$. The results are shown in Table III, while all execution videos are available on the website[1]. We report the total system success rate with re-planning, the percentage of correct single transitions, and the success of any visual plan and action plan from start to goal. Framework F-$L_1$ finds at least one visual action plan that makes the correct prediction, however, the execution of the action is not perfect. We therefore observe a lower overall system performance as the re-planning can result in a premature termination.

| Framework | Syst. | Trans. | VFM | APN |
|---|---|---|---|---|
| F-$L_1$ | **80%** | **90%** | **100%** | **100%** |
| F-$L_2$ | 40% | 77% | 60% | 60% |
| F-$L_\infty$ | 24% | 44% | 56% | 36% |

Table III: Results (best in bold) for executing visual action plans on 5 folding tasks (each repeated 5 times). Different metrics are compared.

Finally, a re-planning example is shown in Fig. 4 where a subset of the proposed visual action plans is shown (left). As the goal configuration does not allude to how the sleeves are to be folded, the LSR suggests all paths it identifies. After the first execution, the re-planning (right) generates in a single plan that leads from start to goal state.

## VII. Conclusions and Future Work

We proposed to build a Latent Space Roadmap which is a graph-based structure in a low-dimensional latent space capturing the latent transition dynamics in a data-efficient manner. Our method consists of a Visual Foresight Module, generating a visual plan from given start and goal states, and an Action Proposal Network, predicting the corresponding action plan. We showed the effectiveness of our method on a simulated box stacking task as well as a T-shirt folding task, requiring deformable object manipulation and performed with a real robot. As future work, we plan to extend the scope of the LSR to more domains such as Reinforcement Learning and validate the approach on a wider set of tasks.

[1]https://visual-action-planning.github.io/lsr/

## References

[1] S. J. Rosenschein, “Formal theories of knowledge in ai and robotics,” *New generation computing*, vol. 3, no. 4, pp. 345–357, 1985.

[2] S. Thrun, D. Fox, and W. Burgard, “Probabilistic methods for state estimation in robotics,” in *Workshop SOAVE*, vol. 97, 1997, pp. 195–202.

[3] T. Lozano-Pérez and L. P. Kaelbling, “A constraint-based method for solving sequential manipulation planning problems,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE, 2014, pp. 3684–3691.

[4] L. P. Kaelbling and T. Lozano-Pérez, “Integrated task and motion planning in belief space,” *Int. J. Robot. Res.*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

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

[6] S. M. LaValle, *Planning Algorithms*. Cambridge U.K.: Cambridge University Press, 2006.

[7] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, “State representation learning for control: An overview,” *Neural Net.*, vol. 108, pp. 379–392, 2018.

[8] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *Int. Conf. Learn. Represent.*, 2015.

[9] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *Int. Conf. Mach. Learn.*, 2014, pp. 1278–1286.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[11] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, “Adversarially learned inference,” in *Int. Conf. Learn. Represent.*, 2017.

[12] B. Ichter and M. Pavone, “Robot Motion Planning in Learned Latent Spaces,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2407–2414, 2019.

[13] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, “Universal planning networks,” in *Int. Conf. Mach. Learn.*, 2018.

[14] A. Wang, T. Kurutach, P. Abbeel, and A. Tamar, “Learning robotic manipulation through visual planning and acting,” in *Robotics: Science and Systems*, 2019.

[15] 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,” *arXiv preprint arXiv:2003.08974*, 2020.

[16] 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, no. 34, 1996, pp. 226–231.

[17] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring Network Structure, Dynamics, and Function using NetworkX,” in *Python in Science Conf.*, 2008, pp. 11–15.

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.