Ensemble Latent Space Roadmap for Improved Robustness in Visual Action Planning

Martina Lippi*[1], Michael C. Welle*[2], Andrea Gasparri[1], Danica Kragic[2]

***Abstract*— Planning in learned latent spaces helps to decrease the dimensionality of raw observations. In this work, we propose to leverage the ensemble paradigm to enhance the robustness of latent planning systems. We rely on our Latent Space Roadmap (LSR) framework, which builds a graph in a learned structured latent space to perform planning. Given multiple LSR framework instances, that differ either on their latent spaces or on the parameters for constructing the graph, we use the action information as well as the embedded nodes of the produced plans to define similarity measures. These are then utilized to select the most promising plans. We validate the performance of our Ensemble LSR (ENS-LSR) on simulated box stacking and grape harvesting tasks as well as on a real-world robotic T-shirt folding experiment.**

I. Introduction

Using raw observations, such as images, for planning in dynamic environments or for tasks such as garment folding may alleviate the need for defining the system state explicitly. Generating plans over the original images is also helpful when displaying the robot state or planned actions to the human operator. However, the high dimensionality of raw observations challenges the effectiveness of classical planning methods [1]. To mitigate this, representation learning can be used to extract compact information from high-dimensional data, enabling the use of classical techniques. Based on this principle, we proposed in [2], [3] a Latent Space Roadmap (LSR) framework, which relies on deep neural networks to learn a low-dimensional latent space and a roadmap in this space to capture state transitions and realize planning.

While utilizing learned models for representation is generally advantageous, a potential drawback lies in the generation of unreliable plans. This arises, for instance, when training involves insufficiently diverse or non-representative data samples. In this work, we aim to enhance the robustness of latent space planning by exploiting the ensemble paradigm [4]: several models are collected and their plans are combined based on similarity measures. To this aim, we take into account both the sequence of actions and the composition of transited nodes in the latent space. In line with majority voting ensemble approaches [5], we select the plans that are the most similar to the others. We name the resulting framework, shown in Fig. 1, Ensemble LSR (ENS-LSR). In detail, our contributions are: *i)* The design of a novel ensemble algorithm for latent space planning along with the definition of appropriate similarity measures. *ii)* Performance comparison with respect to the state-of-the-art Hallucinative Topological Memory (HTM) method [6] and to our previous works [2], [3]. *iii)* Ablation study and validation on multiple tasks, including a new simulated agricultural task and a real-world manipulation task with deformable objects. *iv)* Release of datasets, code, and videos for real-world experiments on the project website[1]. While our focus is on the LSR framework, the proposed algorithm is easily adaptable to any latent space planning method.

*These authors contributed equally (listed in alphabetical order).
[1]Roma Tre University, Rome, Italy {*martina.lippi,andrea.gasparri*}@*uniroma3.it*
[2]KTH Royal Institute of Technology Stockholm, Sweden, {*mwelle,dani*}@*kth.se*

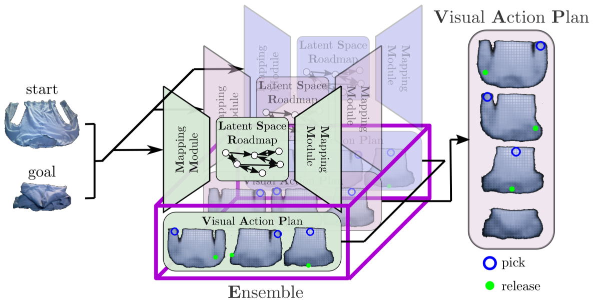

Fig. 1: Depiction of the ENS-LSR framework for a folding task.

II. Related Work

We shortly survey approaches on visual planning and ensemble methods.

A. *Visual planning*

There are several methods that generate plans directly from images. Long-Short Term Memory blocks are used in [7] to generate a video prediction model, which is then integrated into a Model Predictive Control (MPC) framework to generate visual plans, while Generative Adversarial Networks models are employed in [8] to produce visual foresight plans for rope manipulation. However, as mentioned before, latent spaces can reduce the high dimensionality of the image space. For instance, an RRT-based algorithm in the latent space with collision checking is adopted in [9], while interaction features conditioned on object images are learned and integrated within Logic-Geometric Programming for planning in [10]. Additionally, model-free Reinforcement Learning (RL) in the off-policy case combined with auto-encoders is explored in [11], while RL combined with graph structures within the latent space is investigated in [12], where observations are encoded as nodes in the graph and the connectivity is learned from sequence data. The latter approach is extended by the HTM method presented in [6], where a conditional Variational Auto Encoder (VAE) is

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

introduced to *hallucinate* samples given the domain context. Despite the dimensionality reduction, the approaches above typically require a large amount of data. Instead, the LSR framework realizes visual planning in a data-efficient manner by leveraging a contrastive loss to structure the latent space.

B. Ensemble methods

While the ensemble principle has primarily been utilized within the machine learning community [13], it has also been implemented in robotic planning to combine multiple planning algorithms or models and improve the overall system performance. For instance, in [14], multiple planners, working under diverse assumptions, are executed in parallel and combined. An ensemble selection is then made based on learned priors on planning performance. In [15], the combination of multiple heuristics in greedy best-first search planning is investigated. An online ensemble learning method based on different predictors is proposed instead in [16] to achieve an accurate robot forward model that can be beneficial for imitation behavior. The study in [17] adopts an ensemble framework with neural networks to mitigate the error in stereo vision systems and performs planning for manipulation. Finally, several applications of the ensemble paradigm to RL approaches can be found in the literature, such as in [18], [19], [20]. In detail, the ensemble method in [18] combines the value functions of five different RL algorithms, the approach in [19] dynamically interpolates between rollouts with different horizon lengths, while the study in [20] proposes ensemble-based weighted Bellman backups. However, to the best of our knowledge, none of the above approaches is able to realize visual action planning.

III. Preliminaries and Problem Formulation

A. Visual Action Planning

Let $\mathcal{O}$ be the set of all possible observations and $\mathcal{U}$ the set of all feasible actions of the system.

Definition 1: Given start O_s and goal O_g observations, a Visual Action Plan (VAP) $P = (P^o, P^u)$ consists of a visual plan $P^o = (O_s = O_1, O_2, \cdots, O_N = O_g)$, that contains a sequence of N observations showing intermediate states from start to goal, and an action plan $P^u = (u_1, u_2, ..., u_{N-1})$, that provides the respective actions u_i to transition from O_i to O_{i+1}, $\forall i \in \{1, ..., N-1\}$.
Note that in general many plans can feasibly transition the system from start to goal, i.e., VAPs may not be unique.

B. Latent Space Roadmap-based system

The LSR framework relies on a dataset $\mathcal{T}_o$ consisting of tuples (O_i, O_j, ρ), where O_i and O_j are two observations and ρ represents the action information between them. Specifically, $\rho = \{a, u\}$ comprises a binary indicator variable a, which indicates if an action has taken place ($a = 1$) or not ($a = 0$), along with the respective action specification u (which is meaningful only in case $a = 1$). This allows us, in case $a = 0$, to model task-irrelevant variations in observations not caused by actions, such as changes in lighting conditions. Without loss of generality, we consider that the observations in the dataset are numbered and we refer to i in O_i as its index. In the following, we first define the LSR system and then introduce details on its realization and use.

Definition 2: A Latent Space Roadmap-based system S-LSR= $\{\xi, \mathcal{G}, \omega\}$ consists of a latent mapping function ξ, that maps observations to a low dimensional structured latent space, $\xi : \mathcal{O} \rightarrow \mathcal{Z}$, a Latent Space Roadmap $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, that is a graph structure composed of nodes $\mathcal{V}$ and edges $\mathcal{E}$, and an observation generator function ω, that maps latent representations $z \in \mathcal{Z}$ into observations, $\omega : \mathcal{Z} \rightarrow \mathcal{O}$.
We enclose the two functions ξ and ω in a module called Mapping Module (MM). Note that changing either the MM or the LSR produces different S-LSR models. The generic S-LSR$_i$ is defined as S-LSR$_i = \{\xi_i, \mathcal{G}_i, \omega_i\}$. For example, three S-LSRs are depicted in different colors in Fig. 1.
S-LSR implementation: Briefly, based on a dataset $\mathcal{T}_o$, the mapping functions ξ and ω are realized with an encoder-decoder based architecture where a contrastive loss term using the action indicator variable a is introduced [2]. The roadmap $\mathcal{G}$ is then built in the latent space by *clustering* and connecting the latent states associated with the training dataset: each cluster is associated with a node, and edges are constructed if actions exist in the dataset to transition among the respective nodes. As in the action averaging baseline in [2], each edge is additionally endowed with the average action among the corresponding actions of the nodes. Following the steps in [2, Algorithm 2], the LSR building mainly depends on a single parameter, $c_{\max}$, that is an upper bound on the number of weakly connected components of the graph, preventing fragmentation of the graph.
Planning procedure: Given an S-LSR and start (O_s) and goal (O_g) observations, we map them to the structured latent space $\mathcal{Z}$ via ξ and retrieve the closest nodes in the graph $\mathcal{G}$ with centroids in z_s and z_g, respectively. Then, the shortest paths from z_s to z_g are computed. These paths provide both the sequences of latent states, referred to as latent plans $P^z = (z_s = z_1, ..., z_N = z_g)$, and respective actions P^u, retrieved from the graph edges. The latent plans are finally decoded into visual plans P^o using ω. We re-iterate that multiple shortest paths can be found from z_s to z_g.

C. Problem Formulation

Given a dataset $\mathcal{T}_o$, the correctness of plans suggested by an S-LSR mainly depends on [21]: *i)* how well the latent space is structured, i.e., if states are properly separated, and *ii)* how well the LSR is built, i.e., if nodes are representative of the underlying system states and if they are correctly connected according to the possible actions. Both points are influenced by the choice of the respective hyperparameters, i.e., given the same dataset and start and goal observations, two S-LSRs may provide distinct plans, which also differ in terms of correctness, depending on the used hyperparameters. Among the hyperparameters, we include the constitution of the overall dataset in the validation and training parts. Our objective is to design a robust system that can perform visual action planning while mitigating high variance in the results arising from different parameters of the S-LSR.

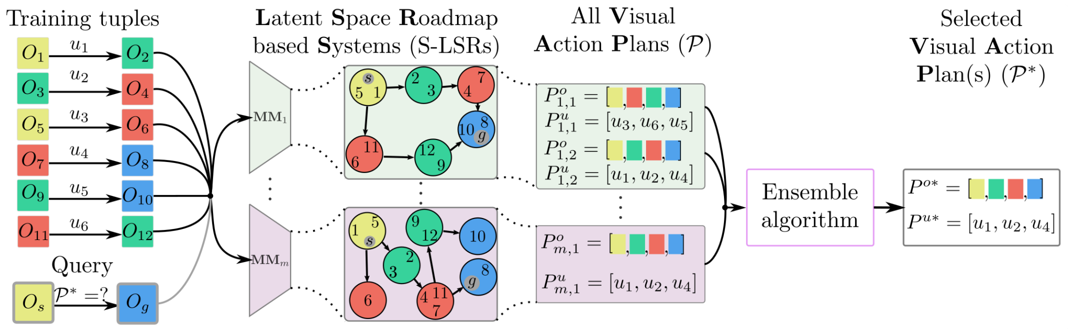
Training tuples
Latent Space Roadmap based Systems (S-LSRs)
All Visual Action Plans ($\mathcal{P}$)
Selected Visual Action Plan(s) ($\mathcal{P}^*$)
O_1 u_1 O_2
O_3 u_2 O_4
O_5 u_3 O_6
O_7 u_4 O_8
O_9 u_5 O_{10}
O_{11} u_6 O_{12}
Query
O_s $\mathcal{P}^*=?$ O_g
MM$_1$
MM$_m$
$P^o_{1,1}=$
$P^u_{1,1}=[u_3,u_6,u_5]$
$P^o_{1,2}=$
$P^u_{1,2}=[u_1,u_2,u_4]$
$P^o_{m,1}=$
$P^u_{m,1}=[u_1,u_2,u_4]$
Ensemble algorithm
$P^{o*}=$
$P^{u*}=[u_1,u_2,u_4]$

Fig. 2: Overview of our ensemble approach. The training tuples are mapped into m separate latent spaces, where individual LSRs are built. The colors of the observations depict their underlying state. Given start and goal observations (bottom left), these are mapped into the latent spaces (grey circles); all the possible VAPs are extracted from the m S-LSRs and a selection is made by the ensemble algorithm.

IV. Ensemble LSR Algorithm

To solve the above problem, we propose to exploit an *ensemble* of m S-LSRs that, as a whole, allow for robustness against outliers. We refer to the ensemble model as ENS-LSR, i.e., ENS-LSR $=\{\text{S-LSR}_1,...,\text{S-LSR}_m\}$. The basic idea is that, given a start and a goal observation, different visual action plans are returned by the S-LSRs and a selection is made among them based on their similarity.

Let $\mathcal{P}_i$ be the set of q_i VAPs produced by the S-LSR i given start O_s and goal O_g observations, i.e., $\mathcal{P}_i=\{P_{i,1},\dots,P_{i,q_i}\}$, where $P_{i,j}$ is the VAP j generated by the S-LSR$_i$. We define the set of all potential VAPs, from O_s to O_g, associated with the ensemble model as the union of the sets $\mathcal{P}_i$, $\forall i\in\{1,...,m\}$, i.e., $\mathcal{P}=\mathcal{P}_1\cup\mathcal{P}_2\cup...\cup\mathcal{P}_m$. Our objective is to select the subset of VAPs $\mathcal{P}^*\subseteq\mathcal{P}$ to provide as output by the ensemble model.

An illustrative overview of the method is provided in Fig. 2 where multiple S-LSRs composing the ensemble model are depicted. The colors of the observations represent their underlying states, e.g., observations with indices 1 and 5 represent the same state in this setting. The figure shows how, starting from the same training tuples, different latent spaces are produced by the mapping modules in ENS-LSR, and different LSRs are built. Hence, given start and goal observations (bottom left), different visual action plans are returned by the S-LSRs. A selection is made by the ensemble algorithm based on their similarity. More specifically, in line with majority voting rule approaches in ensemble learning [5], we propose to select the VAPs that are the most similar to each other, thus removing outlier ones. This implies that, as a first step, appropriate similarity measures must be defined. Then, an algorithm is designed to select the paths based on these measures. In the remainder of the section, we address these two steps. Note that a naive ensemble approach could be to simply propose all plans suggested by all models in ENS-LSR, i.e., $\mathcal{P}\equiv\mathcal{P}^*$. However, in this way no outlier rejection is made and wrong paths can be possibly provided as output (see Sec. V for more details).

A. Similarity measures

We identify two main ways to evaluate the similarity between VAPs: *i)* with respect to actions in the action plans, and *ii)* with respect to node compositions in the latent plans.

Action similarity. As recalled in Sec. III-B, the action plans are obtained by concatenating the average actions contained in the edges of the respective latent plans. Let $P^u_{i,j}$ be the action plan of the j th VAP produced by the S-LSR$_i$. We evaluate the similarity between two action plans $P^u_{i,j}$ and $P^u_{k,l}$ by computing the cosine similarity which captures if the vectors point towards the same direction. To compute this similarity, we preprocess the action plans as follows: first, we collapse the actions of the plans in one dimension (to compare all coordinates), and then, we pad the shortest with zeros to equal the lengths of the sequences (to compare plans with different lengths). For instance, in case of a 2D pick and place action, the action variable can be written as $u=(p,r)$, where $p=(p_x,p_y)$ and $r=(r_x,r_y)$ represent the pick and release positions, and its collapsed version is given by $\bar{u}=(p_x,p_y,r_x,r_y)$. We denote the vector obtained by preprocessing the action plan $P^u_{i,j}$ as $\bar{P}^u_{i,j}$. Therefore, the action similarity measure s^u is obtained as

$$s^u(\bar{P}^u_{i,j},\bar{P}^u_{k,l}):=\frac{1}{2}\left(1+\bar{P}^u_{i,j}\bar{P}^u_{k,l}/\|\bar{P}^u_{i,j}\|\,\|\bar{P}^u_{k,l}\|\right), \qquad (1)$$

with $i\neq k$, which is in the range $[0,1]$ and reaches 1 when the plans are the same, and 0 when they are maximally dissimilar, i.e., they point in opposite directions. Further baselines for the action similarity measure have been validated in Sec. V, which are outperformed by (1).

Node similarity. As recalled in Sec. III-B, the nodes of the i th LSR are obtained by clustering the latent states associated with the training observations. We endow each node with the set of indices of the respective training observations and refer to it as node *composition*. For instance, in Fig. 2, the node compositions of the green nodes in the top LSR are $\{2,3\}$ and $\{12,9\}$. We exploit the simple observation that, if two LSRs have similar node compositions, they will likely produce plans in agreement. Let $P^z_{i,j}$ be the latent plan j of S-LSR$_i$ and $\bar{P}^z_{i,j}$ be the collection of indices associated with the latent plan, i.e., it is the union of the compositions of the nodes in the plan. We define the node similarity measure s^n for two plans $P^z_{i,j}$ and $P^z_{k,l}$ as their Jaccard similarity, i.e., as the ratio of the cardinality of the intersection of the two sets to the cardinality of the union:

$$s^n(\bar{P}^z_{i,j},\bar{P}^z_{k,l}):=|\bar{P}^z_{i,j}\cap\bar{P}^z_{k,l}|/|\bar{P}^z_{i,j}\cup\bar{P}^z_{k,l}|, \qquad (2)$$

with $i\neq k$ and $|\cdot|$ the cardinality of the set $(\cdot)$. This measure is in the range $[0,1]$ and reaches 1 when the plans are the same and 0 when they are maximally dissimilar (no intersection among the plans). Note that the above measure can deal by construction with plans having different lengths.

We define the overall similarity measure s between two VAPs $P_{i,j}$ and $P_{k,l}$ as the sum of the action and the node similarity measures, i.e., $s = s^u + s^n$. Note that both s^u and s^n are in the interval $[0,1]$, making them comparable.

B. Ensemble Latent Space Roadmap Algorithm

Algorithm 1 shows the proposed Ensemble LSR algorithm. It takes as input the set of m S-LSRs composing the ensemble ENS-LSR, as well as the start O_s and goal O_g observations, and returns the most suited visual action plan(s), i.e., $\mathcal{P}^*$, from all the possible ones, i.e., $\mathcal{P}$. The basic idea is to compute a *cumulative comparison score* $c_{i,j}$ for each VAP $P_{i,j}$, based on both action and node similarity measures with respect to the other VAPs, and subsequently select the ones with highest scores. In particular, the score $c_{i,j}$ is obtained by considering, for each S-LSR$_k$ with $k \neq i$, the respective VAP with highest overall similarity measure to $P_{i,j}$ and then by summing up all these highest measures $\forall k \in \{1,...,m\}, k \neq i$. Note that, for each S-LSR$_k$, with $k \neq i$, we consider only one VAP in the computation of the score to ensure that all S-LSRs have equal relevance in that calculation, i.e., to prevent that S-LSRs with more VAPs than others have greater influence in the score than the latter.

At start, the set $\mathcal{P}$ composed of all VAPs, from O_s to O_g, produced by the m S-LSRs is computed (line 1) and an empty set $\mathcal{C}$ is initialized (line 2) to store the cumulative comparison scores. Then, the algorithm iterates over the VAPs of each S-LSR$_i$ (line 3-4). For each $P_{i,j}$, the corresponding action $P^u_{i,j}$ and latent $P^z_{i,j}$ plans are preprocessed as previously described (lines 5 and 6) and the cumulative comparison score $c_{i,j}$ is initialized to zero (line 7). At this point, a comparison is made with the plans produced by the other S-LSRs. Therefore, an iteration is made over the sets of VAPs $\mathcal{P}_k$, with $k \neq i$, obtained by the other S-LSRs and, for each k, an empty set $\mathcal{S}$ is initialized to store the similarity measures between $P_{i,j}$ and the VAPs in $\mathcal{P}_k$. For each VAP $P_{k,l}$ in $\mathcal{P}_k$, the corresponding preprocessed action $\bar{P}^u_{k,l}$ and latent $\bar{P}^z_{k,l}$ plans are obtained (lines 11-12) and their action s^u and node s^n similarities with the plans $\bar{P}^u_{i,j}$ and $\bar{P}^z_{i,j}$ are computed (lines 13-14) according to (1) and (2), respectively. The overall similarity between the VAPs $P_{i,j}$ and $P_{k,l}$ is calculated by summing the individual similarities s^u and s^n and is stored in the set $\mathcal{S}$ (line 15). Once all the overall similarities with the VAPs of the S-LSR$_k$ have been computed, the maximum one is added to the comparison score variable $c_{i,j}$ (line 16). The latter score is complete when all S-LSRs different from i have been analyzed, and is then added to the set $\mathcal{C}$ (line 17). Once all the cumulative comparison scores have been computed, the set of indices $\mathcal{I}$ of the VAPs with the highest scores is extracted (line 18), i.e., $\mathcal{I}^* = \arg\max_{i,j} \mathcal{C}$, and the corresponding VAPs $\mathcal{P}^*$ are selected as output (line 19).

Algorithm 1 Ensemble LSR Planning

```
Require: ENS-LSR = $\{$S-LSR$_1$, ..., S-LSR$_m\}$, $O_s$, $O_g$
 1: $\mathcal{P} \leftarrow$ Compute-VAPs (ENS-LSR, $O_s, O_g$)
 2: $\mathcal{C} \leftarrow \{\}$
 3: for each $\mathcal{P}_i \in \mathcal{P}$ do
 4:     for each $P_{i,j} \in \mathcal{P}_i$ do
 5:         $\bar{P}^u_{i,j} \leftarrow$ Preprocess-action-plan($P_{i,j}$)
 6:         $\bar{P}^z_{i,j} \leftarrow$ Preprocess-latent-plan($P_{i,j}$)
 7:         $c_{i,j} \leftarrow 0$
 8:         for each $\mathcal{P}_k \in \mathcal{P}$, with $k \neq i$ do
 9:             $\mathcal{S} = \{\}$
10:             for each $P_{k,l} \in \mathcal{P}_k$ do
11:                 $\bar{P}^u_{k,l} \leftarrow$ Preprocess-action-plan($P_{k,l}$)
12:                 $\bar{P}^z_{k,l} \leftarrow$ Preprocess-latent-plan($P_{k,l}$)
13:                 $s^u \leftarrow$ Action-sim($\bar{P}^u_{i,j}, \bar{P}^u_{k,l}$)    [eq. (1)]
14:                 $s^n \leftarrow$ Node-sim($\bar{P}^z_{i,j}, \bar{P}^z_{k,l}$)    [eq. (2)]
15:                 $\mathcal{S} \leftarrow \mathcal{S} \cup \{s^u + s^n\}$
16:             $c_{i,j} \leftarrow c_{i,j} + \max(\mathcal{S})$
17:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_{i,j}\}$
18: $\mathcal{I}^* \leftarrow \arg\max_{i,j}(\mathcal{C})$
19: $\mathcal{P}^* \leftarrow$ Select-VAPs($\mathcal{P}, \mathcal{I}^*$)
    return $\mathcal{P}^*$
```

V. Simulation results

We validate our ENS-LSR on two simulation tasks shown in Fig. 3 and realized with Unity [22]: a box stacking task, introduced in [2], and a grape harvesting one. The datasets consist of 2500 and 5000 tuples, respectively, and are available on the website[1]. Actions are expressed as pick and place operations. In the stacking task, 4 boxes with similar textures are stacked in a 3×3 grid and are moved individually. Varying light conditions and positioning noise (up to 15%) are present as task-irrelevant factors of variation. All the actions are reversible in this task, thus undirected LSRs are built. The harvesting task represents a small-scale harvesting setup with 4 grape bunches and consists of 8 cells: 4 in the box and 4 on the vine. Bunches must be moved individually and can only be released inside the box, i.e., a bunch cannot be moved from box to vine or from vine to vine. This implies that the LSRs are directed graphs.

As task-irrelevant factors of variation, we introduce different lighting conditions, bunch scales ($\pm 10\%$), orientations in the box ($\pm 180°$), bunch models differing in the number of grapes as well as positional noise ($\pm 12.5\%$). Compared to the stacking task, visual action planning for the harvesting case is significantly more challenging due to several factors of variations and irreversible actions. In both tasks, the system state is given by the arrangement of the objects in the cells. The simulation setups aim to address these questions:

Q1) What benefits does ENS-LSR offer over individual S-LSRs using different MMs and the HTM [6] method?
Q2) Can ENS-LSR simplify the tuning of the LSR building parameter $c_{\max}$?
Q3) What is the ensemble performance when alternative similarity measures are employed?

We employ the same architectures and parameters as in [2], [3] and quality measure *% all*, which evaluates if all VAPs are correct, using 1000 random start and goal observations

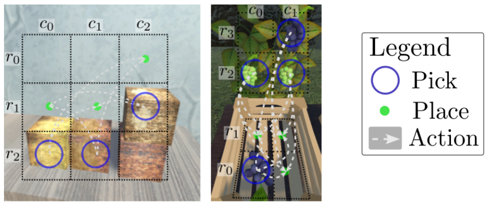

Fig. 3: Box stacking (left) and grape harvesting (right) tasks.

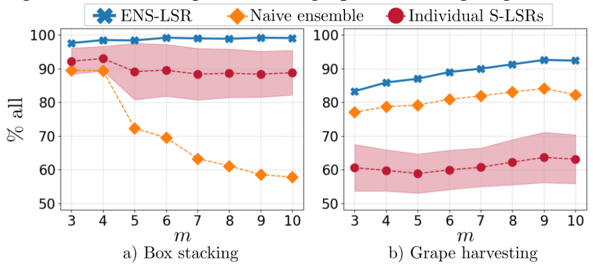

Fig. 4: Comparison of ENS-LSR (in blue) with the individual models (in red) and the naive ensemble approach (in orange).

from holdout datasets. We access the true system states for evaluation only. For harvesting, path feasibility is also checked, e.g., bunches cannot be required to move from box to vine. Unless specified otherwise, we set $c_{\max} = 20$. Example VAPs are shown in the accompanying video.

A. ENS-LSR via different mapping modules

To answer Q1), we consider for each task an ensemble of ten S-LSRs obtained with different MMs. These are generated by randomly selecting 85% of the training data. Results in terms of *% all* are reported in Fig. 4 (stacking on the left and harvesting on the right) as the number m of S-LSRs increases from three to ten. Specifically, the figure shows the performance achieved by ENS-LSR (in blue), individual S-LSRs (in red with means and variances represented as dots and shadows, respectively), and the naive ensemble approach (in orange) which outputs all possible VAPs, i.e., $\mathcal{P}^* = \mathcal{P}$. In both tasks, ENS-LSR significantly outperforms the other methods. For the box stacking task, the naive approach results in decreasing performance as the number of S-LSRs increases, which is motivated by the fact that the higher m, the higher the likelihood that at least one wrong plan is suggested. The individual models provide high variance and report a slight performance drop with the addition of the fifth S-LSR, which only reaches 73.7%. In contrast, our ENS-LSR is not affected by the addition of this S-LSR and already scores 97.6% with $m = 3$ and achieves 99.1% with $m = 10$. For the harvesting task, the individual S-LSRs only achieve about 60% on average for all m since they are often unable to find a path. This is overcome by the naive method, where only one of the S-LSRs needs to find a path, reaching 77.1% with $m = 3$. However, the naive approach's performance decreases as the number of S-LSRs increases and an underperforming model is included (at $m = 10$). ENS-LSR not only outperforms the other methods, achieving 83.3% with $m = 3$ and 92.6% with $m = 9$, but is also robust when low-performing models are included, obtaining 92.4% with $m = 10$.

Comparison with HTM. For the HTM method, we use the same setting as in [6] and tailor it to the box stacking dataset

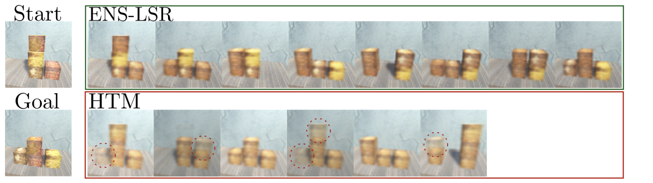

Fig. 5: Visual plans obtained by ENS-LSR (top row) and HTM (bottom row) for the box stacking task given start and goal (left).

(with empty scene as context, 5000 hallucinated samples and one step training trajectories). Surprisingly, over 1000 test cases, HTM fails to produce any correct paths. This is due to HTM often suggesting actions that violate stacking rules and generating unfeasible states with, for instance, an excessive number of boxes. Figure 5 reports an example of visual plans produced by ENS-LSR with $m = 3$ (top) and HTM (bottom) given start and goal observations (on the left). While ENS-LSR provides a correct plan, HTM leads to both unfeasible states (circles mark partially transparent boxes) and illegal actions. Further examples can be found on the website. Note that these results do not undermine the effectiveness of HTM in general, but only show its unsuitability for high-level task planning with limited training data as in our setup.

B. ENS-LSR via different LSR hyperparameters

We examine Q2) with a group of ten S-LSRs that differ in LSR hyperparameter $c_{\max}$, while utilizing the same MM. Table I summarizes the results for both simulation tasks (stacking on top and harvesting in the bottom) with $c_{\max}$ ranging from 1 to 90. The performance index *% all* and the percentage of times a VAP is found, denoted as *% $\exists$ path*, are reported for ENS-LSR (second column) and the individual S-LSRs (third to last column). Results show that ENS-LSR outperforms any individual S-LSR in both tasks for both metrics. Specifically, for the stacking task, ENS-LSR achieves *% all* equal to 98.8%, outperforming the best individual model (obtained with $c_{\max} = 10$) by 2.4%. Note that high $c_{\max}$ values result in low performance as they lead to the construction of more disconnected graphs. Similarly, the *% $\exists$ path* metric reaches 100% with ENS-LSR, while for the individual S-LSRs, it starts from 100% in case of a connected graph with $c_{\max} = 1$ and then decreases as $c_{\max}$ increases. For the harvesting task, similar trends can be observed: ENS-LSR consistently outperforms individual S-LSRs by at least 10% for *% all* (reaching 69.6%) and finds a path in the majority of cases ($\exists$ *path*= 98.8%), while with the individual S-LSRs, the higher $c_{\max}$, the lower the capability of finding paths, reaching $\exists$ path= 38.7% with $c_{\max} = 90$. Note that while $c_{\max} = 1$ enforces one connected component, the respective S-LSR only achieves $\exists$ *path*= 98.1% as it incorrectly assumes the same underlying state for the start and goal states in 1.9% cases.

C. Similarity measure comparison

For Q3), we analyze additional action and node similarity measures. Regarding the action measures, we include *i)* a simple baseline based on the Euclidean distance, that is $s^u_{eucl}(P^u_{i,j}, P^u_{k,l}) = -\|P^u_{i,j} - P^u_{k,l}\|$, which is 0 when the plans are the same, and negative otherwise, and where plans with

Stacking	ENS-LSR	$c_{\max}=1$	$c_{\max}=10$	$c_{\max}=20$	$c_{\max}=30$	$c_{\max}=40$	$c_{\max}=50$	$c_{\max}=60$	$c_{\max}=70$	$c_{\max}=80$	$c_{\max}=90$
% all	**98.8**	93.4	96.4	96.0	94.5	91.9	88.5	84.0	80.3	78.0	75.4
% ∃ path	**100.0**	**100.0**	97.6	96.9	95.3	92.9	89.7	85.2	81.4	79.1	76.5

Harvesting	ENS-LSR	$c_{\max}=1$	$c_{\max}=10$	$c_{\max}=20$	$c_{\max}=30$	$c_{\max}=40$	$c_{\max}=50$	$c_{\max}=60$	$c_{\max}=70$	$c_{\max}=80$	$c_{\max}=90$
% all	**69.6**	11.0	57.6	56.7	54.8	51.9	48.2	44.6	40.9	39.2	38.5
% ∃ path	**98.8**	98.1	76.5	65.1	59.4	54.4	48.8	44.7	41.0	39.3	38.7

TABLE I: Results achieved by ENS-LSR and individual models with different $c_{\max}$ values for both tasks. Best results in bold.

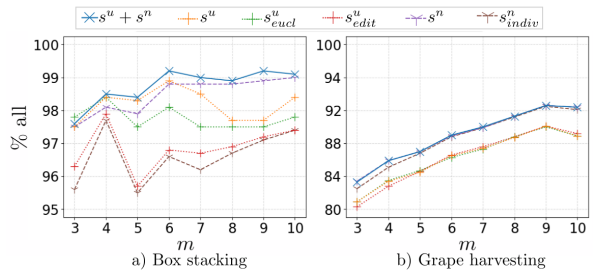

Fig. 6: Comparison with different similarity measures on the box stacking (left) and grape harvesting (right) tasks.

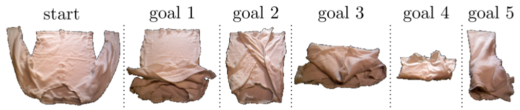

Fig. 7: Start and goal configurations for the real-world folding task.

Framework	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
ENS-LSR	**5/5**	**5/5**	**4/5**	**5/5**	3/5
ACE-LSR	4/5	**5/5**	3/5	4/5	**4/5**

TABLE II: Performance results on the T-shirt folding task with ENS-LSR and ACE-LSR. Best results in bold.

different lengths are ignored; and *ii)* the edit similarity [23], denoted as s^u_{edit}, which is based on counting and weighting the number of operations (insertion, deletion, or substitution) needed to transform one plan into the other. To define if two actions u_i and u_j are the same, we require their Euclidean distance to be lower than a threshold, which is 0.5 in our tests. We set costs for insertion and deletion equal to -0.5 and -1, respectively, while for the substitution we use the opposite of the Euclidean distance. For the node similarity, we consider a baseline, denoted as s^n_{indiv}, where the Jaccard similarity between *individual* compositions of nodes in the plans is computed and then aggregated for all nodes, i.e.,

$$s^n_{indiv}(P^z_{i,j}, P^z_{k,l}) = \sum_{t=1}^{N} |C_{i,j,t} \cap C_{k,l,t}|/|C_{i,j,t} \cup C_{k,l,t}|,$$

with $C_{i,j,t}$ and $C_{k,l,t}$ the compositions of the tth node in the plans $P^z_{i,j}$ and $P^z_{k,l}$, respectively. Plans with different lengths are ignored. We compare the performance of ENS-LSR when using each similarity measure individually in line 15 of Algorithm 1. Results are shown in Fig. 6. For the stacking task (on the left), we observe that all similarity measures lead to high performance *% all*, but in general, the ones requiring equal path lengths (i.e., s^u_{eucl} and s^n_{indiv}) perform worse, while the overall similarity measure (i.e., $s^n + s^u$) outperforms all others and remains robust even if any individual similarity measure underperforms. For the harvesting task (on the right), a clear performance difference between node-based similarity measures and action-based ones is observed, with the former outperforming the latter by approximately 3%. This can be motivated by the fact that node-based similarity measures have much more fine-grained information compared to the action-based ones which is beneficial for a more accurate comparison. Finally, the figure shows that $s^n + s^u$ does not lead to any undesirable performance drop for $m = 10$ compared to s^n_{indiv}.

VI. Folding results

We validate the effectiveness of ENS-LSR in a real-world T-shirt folding task as in [3]. This task consists of a start state, with the T-shirt spread on the table, and five goal states, as shown in Fig. 7. Actions are expressed as pick-and-place operations and are not reversible, leading to directed LSRs. All execution videos and the dataset are available on the website[1]. By exploiting the insights from the simulation tasks, we construct an ENS-LSR consisting of ten MMs and three $c_{\max}$ values - specifically, 1, 5, and 10, leading to $m = 30$. At planning time, the ENS-LSR takes the current T-shirt observation and a goal configuration as input and produces one or more VAPs as output $\mathcal{P}^*$. The first action of a plan in $\mathcal{P}^*$ is executed with Baxter robot, as shown in the accompanying video. After each action, a replanning step is made using the current state as start configuration, until the desired goal is reached or no path is found. Note that, since ENS-LSR aims to suggest plans that are all correct, it eliminates the need for human selection as in [2], [3], but simply randomly selects and executes a VAP in $\mathcal{P}^*$. For each goal configuration, we execute five tests as in our previous works and collect the results in Table II,where we compare it with our latest framework [3], i.e., ACE-LSR, implementing a paradigm for dealing with data scarcity. We employ for both models a dataset composed of 562 tuples, which is 50% of the training data used in [2]. Results show that ENS-LSR is generally more robust than ACE-LSR, successfully executing folds 1, 2, and 4 all the times, and achieving overall performance of 88%, which outperforms by 8% the previously top-performing ACE-LSR model. A slight performance decrease is only recorded with Fold 5, where ENS-LSR occasionally misses the final step that involves picking up a small T-shirt edge, as shown on the website[1].

VII. Conclusions

In this work, we presented a novel ensemble algorithm for visual action planning. The method relies on multiple latent space roadmaps-based systems, which can be obtained either by different mapping modules or LSR hyperparameters. Given start and goal observations, a selection of the plans to output is made on the basis of action- and node-based similarity measures. We validated the approach on two simulation tasks and on a real-world folding task as well as compared it with the state-of-the-art HTM method. In future work, we aim to leverage different latent spaces to facilitate the transfer of knowledge across a range of tasks.

References

[1] R. Bellman, "Curse of dimensionality," *Adaptive control processes: a guided tour. Princeton, NJ*, vol. 3, p. 2, 1961.

[2] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic, "Enabling visual action planning for object manipulation through latent space roadmap," *IEEE Trans. Robot.*, 2022.

[3] M. Lippi, M. C. Welle, P. Poklukar, A. Marino, and D. Kragic, "Augment-connect-explore: a paradigm for visual action planning with data scarcity," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 754–761, 2022.

[4] L. Rokach, *Ensemble learning: pattern classification using ensemble methods*. World Scientific, 2019.

[5] D. Ruta and B. Gabrys, "Classifier selection for majority voting," *Information fusion*, vol. 6, no. 1, pp. 63–81, 2005.

[6] K. Liu, T. Kurutach, C. Tung, P. Abbeel, and A. Tamar, "Hallucinative topological memory for zero-shot visual planning," in *Int. Conf. Mach. Learn.*, pp. 6259–6270, 2020.

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

[8] A. Wang, T. Kurutach, P. Abbeel, and A. Tamar, "Learning robotic manipulation through visual planning and acting," in *Robot.: Science and Syst.*, 2019.

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

[10] J.-S. Ha, D. Driess, and M. Toussaint, "Deep visual constraints: Neural implicit models for manipulation planning from visual input," *IEEE Robot. Autom. Letters*, vol. 7, no. 4, pp. 10857–10864, 2022.

[11] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *AAAI Conf. Artif. Intell.*, vol. 35, pp. 10674–10681, 2021.

[12] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *Int. Conf. Learn. Represent.*, 2018.

[13] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Front. Comput. Science*, vol. 14, pp. 241–258, 2020.

[14] S. Choudhury, S. Arora, and S. Scherer, "The planner ensemble: Motion planning by executing diverse algorithms," in *IEEE Int. Conf. Robot. Autom.*, pp. 2389–2395, 2015.

[15] G. Röger and M. Helmert, "The more, the merrier: Combining heuristic estimators for satisficing planning," *Int. Conf. Automa. Plan. and Schedul.*, vol. 20, no. 1, pp. 246–249, 2021.

[16] M. Zambelli and Y. Demirisy, "Online multimodal ensemble learning using self-learned sensorimotor representations," *IEEE Trans. Cogn. Developmental Syst.*, vol. 9, no. 2, pp. 113–126, 2017.

[17] A. Shahzad, X. Gao, A. Yasin, K. Javed, and S. M. Anwar, "A vision-based path planning and object tracking framework for 6-dof robotic manipulator," *IEEE Access*, vol. 8, pp. 203158–203167, 2020.

[18] M. A. Wiering and H. van Hasselt, "Ensemble algorithms in reinforcement learning," *IEEE Trans. Syst. Man Cybern., Part B*, vol. 38, no. 4, pp. 930–936, 2008.

[19] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[20] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning," in *Int. Conf. on Mach. Learn.*, pp. 6131–6141, 2021.

[21] 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," *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2022.

[22] Unity Technologies, "Unity." https://unity.com.

[23] P. Bille, "A survey on tree edit distance and related problems," *Theoretical Computer Science*, vol. 337, no. 1, pp. 217–239, 2005.