Transporters with Visual Foresight for Solving Unseen Rearrangement Tasks

Hongtao Wu*, Jikai Ye*, Xin Meng, Chris Paxton, Gregory S. Chirikjian

Abstract—Rearrangement tasks have been identified as a crucial challenge for intelligent robotic manipulation, but few methods allow for precise construction of unseen structures. We propose a visual foresight model for pick-and-place rearrangement manipulation which is able to learn efficiently. In addition, we develop a multi-modal action proposal module which builds on the Goal-Conditioned Transporter Network, a state-of-the-art imitation learning method. Our image-based task planning method, Transporters with Visual Foresight, is able to learn from only a handful of data and generalize to multiple unseen tasks in a zero-shot manner. TVF is able to improve the performance of a state-of-the-art imitation learning method on unseen tasks in simulation and real robot experiments. In particular, the average success rate on unseen tasks improves from 55.4% to 78.5% in simulation experiments and from 30% to 63.3% in real robot experiments when given only 10% of expert demonstrations. Video and code are available on our project website: https://chirikjianlab.github.io/tvf/

I. INTRODUCTION

Prospection enables humans to imagine effects of actions [1]. It allows humans to learn multiple tasks from very few examples and generalize to unseen tasks efficiently by combining skills that they have honed in other contexts. If robots are to become efficient in learning new tasks, this ability will be essential.

One version of prospection is visual foresight, which predicts action effects by hallucinating the expected changes in the robot’s observation space. This has been successfully demonstrated in various robotic manipulation tasks [2]–[8]. However, training an accurate and reliable visual dynamics model requires copious amounts of data. Previous work [2], [4], [5] collects a large amount of data for training the dynamics model. In addition, most methods focus on single-task learning with visual foresight [2], [3], [6]–[8]. While some prior work showcases multi-task learning [4], [5], the ability to generalize to unseen tasks which are not present in the training data is still missing. Moreover, the huge combinatorial search space of possible actions makes planning for complex multi-step tasks computationally challenging.

In this work, we first propose a visual foresight (VF) model which predicts the next-step observation based on the current observation and a pick-and-place action. Unlike previous methods which encode actions in the latent space [2], [3], our model exploits the spatial equivariance in vision-based manipulation tasks by encoding the pick-and-place action in the image space. This allows our VF model to learn efficiently and predict accurate next-step observations even with only tens of training data. Secondly, we develop a multi-modal action proposal module by leveraging a state-of-the-art imitation learning method for rearrangement tasks, the Goal-Conditioned Transporter Network (GCTN) [9]. Combining the VF model and the action proposal module with a tree-search algorithm, we propose Transporters with Visual Foresight (TVF), a novel goal-conditioned task planning method for rearrangement tasks that achieves zero-shot generalization to multiple unseen tasks that are structurally similar to the training tasks given only a handful of expert demonstrations.

We perform experiments on both simulation and real robot platforms and compare with the state-of-the-art baseline method GCTN [9]. In simulation experiments, our method achieves an average success rate of 78.5% on 8 unseen tasks given only 10% of expert demonstrations, compared to the 55.4% success rate of GCTN. In real robot experiments, given 30 expert demonstrations, our method achieves an average success rate of 63.3% on 3 unseen tasks, outperforming GCTN which achieves 30.0%.

The key contributions of this work are: (1) a goal-conditioned task planning method which achieves zero-shot generalization to unseen, long-horizon rearrangement tasks; (2) a visual foresight model which is able to learn efficiently from a handful of data; and (3) a multi-modal action proposal module for more versatile action proposal.

II. RELATED WORK

A. Transporter Networks

The most relevant work is probably the Transporter Networks (TN) for efficient learning of tabletop rearrangement...
tasks. Seita et al. [9] build on Transporter Networks and propose Goal-Conditioned Transporter Networks (GCTN) for handling deformable objects. However, both TN and GCTN focus on single-task learning and do not address unseen task generalization. CLIPort [12] extends Transporter Networks to a language-conditioned policy that solves language-specific tabletop tasks. Lim et al. [11] propose Sequence-Conditioned Transporter Networks (SCTN) based on GCTN with sequence conditioning for accomplishing a sequence of tasks in a single rollout. SCTN uses a human oracle to provide different intermediate next-step goal images for GCTN to be conditioned on throughout a rollout. Our multi-task setting focuses on a single task for a single rollout instead of a sequence of tasks in a single rollout. In addition, instead of manually providing multiple intermediate next-step goals during the rollout, in our method, only one single last-step goal is provided and fixed throughout a rollout.

B. Visual Foresight

Research on visual foresight (VF) has become popular in recent decades [2]–[4], [6], [14]–[16]. Finn and Levine [2] use a convolutional LSTM architecture for video prediction and demonstrate on a robot pushing task with model predictive control (MPC). Building on [2], Ebert et al. [14] account for occlusion by adding temporal skip-connections to the architecture. In [6], Huang et al. utilize a Mask-RCNN for object segmentation and predict the transformation for each object in the next frame. Kossen et al. [15] discover hidden structures in images using a latent variable model and predict the dynamics in the latent space with a graphical neural network. Minderer et al. [16] also adopt a latent variable model but instead use a keypoint detector for hidden structure extraction. Paxton et al. [3] and Hoque et al. [4] train a visual dynamics model with a huge amount of data to infer the next-step image after taking an action and plan based on evaluating the value of the predicted image. In contrast to requiring copious amounts of data for training, our method learns the visual dynamics model efficiently with only a handful of data.

C. Task Planning

Task planning has been popular for decades in robotics [3]–[5], [7], [17]–[25]. Task and motion planning (TAMP) [17]–[19] solves complex tasks by integrating discrete high-level planning with continuous low-level planning. These methods generally rely on full knowledge about the world and state estimation of objects, although learning has been applied to weaken these assumptions recently [21], [24]. Another line of work uses modular perception frameworks to obtain object states [7], [20] or latent representations of objects [23] and proposes actions with a single-task policy network. Our method differs from these methods by encoding object states in a visual dynamics model instead of relying on state estimation.

D. Multi-Task and Meta Learning

There is a growing interest in multi-task learning [11], [26]–[30] and meta-learning [31]–[33]. Multi-task learning has been studied in reinforcement learning settings with shared multi-task policies [28], [29] and gating networks [27]. Meta-learning [31]–[33] and task-to-task transfer [30] aim to learn a policy which is able to perform well on a new task given only a few demonstrations of that task. In contrast to learning a policy that performs multiple training tasks or a new task given more training data of that task, our goal is to learn a policy that is able to perform multiple unseen tasks that are structurally similar to the training tasks in a zero-shot manner.

III. PROBLEM FORMULATION

We formulate the planar tabletop rearrangement task as learning a goal-conditioned policy \( \pi \) that maps a current observation \( o_t \) and a goal observation \( o_g \) to an action \( a_t \):

\[
\pi(o_t, o_g) : (o_t, o_g) \rightarrow a_t
\]  

The action is specified by a pair of picking and placing poses \( a_t = (T_{\text{pick}}, T_{\text{place}}) \) where \( T_{\text{pick}}, T_{\text{place}} \in SE(2) \). Similar to [10], the observation in our method is an orthographic top-down view \( o_t \in \mathbb{R}^{H \times W \times 4} \) of the tabletop workspace. Each
pixel corresponds to a vertical column in the workspace. The four channels of $o_t$ are the RGB channels and the height of the column in the workspace. We are able to map every pixel $p = (u, v) \in o_t$ to a picking (or placing) position on the table via camera-to-robot calibration. The picking and placing actions are parameterized as $T_{\text{pick}} = (p_{\text{pick}}, \theta_{\text{pick}})$ and $T_{\text{place}} = (p_{\text{place}}, \theta_{\text{place}})$. The goal observation is the top-down view $o_g \in \mathbb{R}^{H \times W \times 4}$ of the goal scene. To execute $a_t$, the robot first moves to $T_{\text{pick}}$ and lowers its end effector. The gripper is then activated to grasp the object. After the object is grasped, the robot moves upward and then towards $T_{\text{place}}$. After arriving at $T_{\text{place}}$, the robot lowers its end effector and deactivates the gripper to release the object.

Our goal is to learn a policy that generalizes to multiple unseen tasks in a zero-shot manner. To train $\pi$, we assume a small dataset containing expert demonstrations of $M$ different tasks $D = \{\xi_i\}_{i=1}^N$ as provided as training data where $\xi_i$ is an episode of a task. An episode $\xi_i$ of length $T_i$ contains observations and actions of different steps: $\xi_i = \{o_1, a_1, o_2, a_2, \ldots, o_{T_i}, a_{T_i}, o_{T_i+1}\}$ where $o_{T_i+1} = o_g$.

IV. METHODS

In this section, we first introduce the proposed visual foresight (VF) model. We then describe the multi-modal action proposal module developed from Goal-Conditioned Transporter Networks (GCTN) [9]. Finally, we introduce combining the VF model and the action proposal module with a tree-search algorithm for long-horizon task planning.

A. Visual Foresight (VF) Model

Fig. 3 shows the network architecture of our VF model. It takes as input the top-down observation $o_t$ and a pick-and-place action $a_t = (T_{\text{pick}}, T_{\text{place}})$, and outputs the imagined observation of the next step $o_{t+1} \in \mathbb{R}^{H \times W \times 4}$:

$$o_{t+1} = f(o_t, a_t)$$

(2)

$T_{\text{pick}}$ is specified by a binary image $M_{\text{pick}} \in \mathbb{R}^{H \times W}$ which is positive within a square mask centered at $p_{\text{pick}}$ and zero elsewhere. To represent $T_{\text{place}}$, we first rotate $o_t$ by $\Delta \theta = \theta_{\text{place}} - \theta_{\text{pick}}$ about a pivot positioned at $p_{\text{pick}}$. The rotated $o_t$ is then cropped at $p_{\text{pick}}$ with a square of the same size as the mask in $M_{\text{pick}}$. The cropped image is pasted on a zero image at $p_{\text{place}}$ to create $M_{\text{place}} \in \mathbb{R}^{H \times W \times 4}$. A fully convolutional network (FCN) [34] takes as input the concatenated image of $o_t$, $M_{\text{pick}}$, and $M_{\text{place}}$ and outputs $o_{t+1}$. The FCN is a feed-forward neural network composed of convolution and deconvolution blocks with residual connections [35]. Intuitively, our VF model imagines $o_{t+1}$ by “cutting” $o_t$ with a square mask at $p_{\text{pick}}$, the cut by $\Delta \theta$, and “pasting” the cut by overlaying it on top of $o_t$ at $p_{\text{place}}$.

Spatial Equivariance for Efficient Learning. Our VF model achieves high sample efficiency by utilizing a spatially consistent input and introducing inductive biases in the network design. In fact, the dynamics of the tabletop rearrangement problem are $SE(2)$ equivariant; that is, applying a transformation $g \in SE(2)$ on $o_t$ and $a_t$ will lead to an identical transformation on $o_{t+1}$. If we define $x_t = (o_t, a_t)$, the $SE(2)$ equivariance can be written as:

$$f(g \cdot x_t) = g \cdot f(x_t)$$

(3)

where $g \cdot x_t = (g \cdot o_t, g \cdot a_t)$. $\cdot$ indicates the operation on observation $o_t$ which transforms the pixels with $g$; $g \circ a_t \triangleq (g \circ T_{\text{pick}}, g \circ T_{\text{place}})$ and $\circ$ is the group operation for $SE(2)$.

More details on $SE(2)$ equivariance of the dynamics can be found in the supplementary materials on our project page. By using a spatially consistent observation and encoding the action in the image space, we are able to take advantage of the $SE(2)$ equivariance and conveniently implement data augmentation by applying the same rigid transform to $o_t$, $a_t$, and $o_{t+1}$, as shown in prior works [9], [10], [12].

In addition, encoding actions in the image space instead of the latent space as [2], [3] allows us to use an FCN as our network architecture and take advantage of the translational equivariance property of the network [34]. This property is very desirable and has been previously shown to improve learning efficiency in vision-based manipulation [9], [10], [12]. While we have achieved translational equivariance with an FCN in this work, the dynamics for tabletop rearrangement is in general $SE(2)$ equivariant. We leave the investigation on using $SE(2)$ equivariant networks as future work. In addition, the changes of the scene in tabletop rearrangement mostly happen locally in the vicinity of picking and placing positions. By using a square mask positioned locally at $p_{\text{pick}}$ and $p_{\text{place}}$, our VF model intuitively captures this feature – the FCN only needs to attend to the local regions of picking and placing in $o_t$.

B. Multi-Modal Action Proposal

Background. Goal-Conditioned Transporter Networks (GCTN) [9] are a powerful approach for pick-and-place rearrangement manipulation. Their observation space is also an orthographic top-down view of the tabletop workspace as introduced in Sec. III. GCTN takes as input the observation of the current scene $o_t$ and the goal scene $o_g$ and outputs a pick-and-place action $a_t = (T_{\text{pick}}, T_{\text{place}})$. It consists of 4 FCNs. The first FCN takes as input $o_t$ and $o_g$ and outputs a dense action-value map $Q_{\text{pick}} \in \mathbb{R}^{H \times W}$ which correlates with the picking success. The picking position is given
Fig. 4: Multi-Modal Action Proposal. We use K-Means Clustering to identify candidate high-level actions over which we can plan. The pixels highlighted with yellow indicate placing actions with high-value. The green dots in the image after K-Means Clustering are the cluster centers for the three clusters. The three maps showing the placing actions for each cluster.

by $p_{\text{pick}} = \arg\max_{(u,v)} Q_{\text{pick}}((u,v)|o_t, o_g)$. By using a symmetric gripper, e.g., a suction cup gripper, $p_{\text{pick}}$ is set to 0. For the placing action, the orientation space (i.e., $SO(2)$) is discretized into $R$ bins. The final three FCNs generate the placing action-value map $Q_{\text{place}} \in \mathbb{R}^{H \times W \times R}$ which correlates with the placing success. $Q_{\text{place}}(u,v,r)$ indicates the placing success of the action $i = (u, v, 2\pi r / R)$. The placing action is given by $T_{\text{place}} = \arg\max_i Q_{\text{place}}(t|o_t, o_g, p_{\text{pick}})$. More details about GCTN can be found in [9].

Algorithm 1: MultiModalActionProposal($o_t, o_g$)

1. $(T_{\text{pick}}, Q_{\text{place}}) \leftarrow \text{GCTN}(o_t, o_g)$
2. $Q_{\text{place}} \leftarrow \max_{(u,v,\theta)} Q_{\text{place}}(u,v,\theta)$
3. $Q_{\text{max}} \leftarrow \max \{Q_{\text{pick}}((u,v)|o_t, o_g)\}$
4. $\theta(u,v) \leftarrow \arg\max_{\theta} Q_{\text{place}}(u,v,\theta)$
5. $S \leftarrow \emptyset$
6. foreach $(u,v)$ do
7.   if $Q_{\text{place}}(u,v) > \alpha Q_{\text{max}}$ then
8.     append $(u,v)$ to $S$
9. end
10. $S = \text{Top}_N(S)$
11. for $i \leftarrow 1, 2, \cdots, K$ do
12.   $(u_i, v_i) \leftarrow \arg\max_{(u,v)} Q_{\text{place}}(u,v)$
13.   $\theta_i \leftarrow \theta(u_i, v_i)$
14.   $T_{\text{place}}^i \leftarrow (u_i, v_i, \theta_i)$
15. end
16. return $T_{\text{pick}}, T_{\text{place}}^1, T_{\text{place}}^2, \cdots, T_{\text{place}}^K$

Multi-Modality with K-means Clustering. Instead of outputting only one single action as in GCTN [9], we want to explore more actions for multi-modality and versatility when tackling unseen tasks. We propose multiple actions by pairing $T_{\text{pick}}$ generated by maximizing $Q_{\text{pick}}$ with multiple $T_{\text{place}}^i$ generated from the placing action-value map $Q_{\text{place}}$ with K-Means Clustering.

The algorithm is given in Alg. 1. Specifically, given $T_{\text{pick}}$ and $Q_{\text{place}}$ outputted by GCTN, we first find the maximum value of $Q_{\text{place}}$ which we denote as $Q_{\text{max}}$ (Line 2). We then maximize over the rotation channel to get the max rotation map $Q_{\text{place}} \in \mathbb{R}^{H \times W}$ (Line 3). The pixels with a value larger than $\alpha Q_{\text{max}}$ are selected and appended to a list $S$, where $\alpha \in (0, 1)$ is a hyperparameter. We use $\alpha$ to filter out pixels of which the values are not substantial – their corresponding actions are considered bad proposals and will become noise in the clustering. The top $N$ pixel positions within $S$ are selected (Line 11) and used for K-Means Clustering (Line 12) to generate $K$ clusters. For each cluster $A_i$, the pixel position $(u_i, v_i)$ with the maximum value of $Q_{\text{place}}$ plus its corresponding rotation angle $\theta_i$ is used as the place action $T_{\text{place}}^i$ for the cluster. See Fig. 4 for visualization.

Algorithm 2: TreeSearch($o_t, o_g, d_{\text{max}}$)

1. $n_0 \leftarrow [o_t, 0, 0]$
2. $L \leftarrow \emptyset$
3. $L_{\text{prev}} \leftarrow [n_0]$
4. for $i \leftarrow 1, 2, \cdots, d_{\text{max}}$ do
5.   $L_{\text{curr}} \leftarrow \emptyset$
6.   foreach $n$ in $L_{\text{prev}}$ do
7.     $[o_a, d, \tau] \leftarrow n$
8.     $(T_{\text{pick}}, T_{\text{place}}^1, \cdots, T_{\text{place}}^K) \leftarrow \text{MultiModalActionProposal}(o_a, o_g)$
9.     for $k \leftarrow 1, 2, \cdots, K$ do
10.        $o' \leftarrow f(o_a, (T_{\text{pick}}, T_{\text{place}}^k))$
11.        $\tau' \leftarrow \tau \cup \{T_{\text{place}}^k\}$
12.        $n' \leftarrow [o', d + 1, \tau']$
13.        $L_{\text{curr}} \leftarrow L_{\text{curr}} \cup \{n'\}$
14.     $L_{\text{prev}} \leftarrow L_{\text{curr}}$
15.   end
16. return $L$

C. Transporters with Visual Foresight

Combining the VF model (Sec. IV-A) and the multi-modal action proposal module (Sec. IV-B) with a full tree-search algorithm, we propose Transporters with Visual Foresight (TVF). Alg. 2 shows the tree search algorithm. The maximum depth $d_{\text{max}}$ of the tree is a hyperparameter. Each edge in the tree corresponds to an action. Each node $n$ contains the current observation $o$, the depth of the node $d$, and the action sequence $\tau$ which leads the root node $n_0$ to the current node. A typical tree search iteration (Line 7-13) consists of 2 steps: action proposal and node expansion. The action proposal takes as input the observation of the node $o$ and the goal $o_g$ and outputs a picking action and multiple companion placing actions (Line 8). The VF model expands the node by taking each pick-and-place action pair to generate the imagined observation of the next step $o'$ (Line 10). $o'$ is used to construct a new node for a new search iteration (Line 12). Throughout the expansion, we maintain a list $L$ which contains all nodes in the tree.

Alg. 3 shows the algorithm of TVF. After the tree is fully expanded, the value of each node is given by $C - L_1(o, o_g)$ in which $C$ is a positive constant; $L_1(\cdot, \cdot)$ is the mean absolute error; $o$ and $o_g$ are the imagined observation of the node and the goal observation, respectively (Line 5). To bias the policy towards short search paths, the value is further multiplied by a discount factor $\gamma^{d-1}$ which decays with the increase of the depth $d$ as $\gamma \in (0, 1)$. The node with the largest
value is chosen. The robot takes the first action in the action sequence of the node (Line 9), and then replans until the task is accomplished or the step number exceeds the maximum step number.

**Algorithm 3: TVF(o_t, o_g, \(d_{\text{max}}\))**

1. \(L \leftarrow \text{TreeSearch}(o_t, o_g, d_{\text{max}})\)
2. \(v_{\text{max}} = 0\)
3. **foreach** \(n\) in \(L\) do
   4. \([o, d, T] \leftarrow n\)
   5. \(v_{\text{tmp}} \leftarrow \gamma^{d-1}(C - L_1(o, o_g))\)
   6. **if** \(v_{\text{tmp}} > v_{\text{max}}\) **then**
   7. \(v_{\text{max}} \leftarrow v_{\text{tmp}}\)
   8. \(n_{\text{best}} \leftarrow n\)
9. \((T_{\text{pick}}, T_{\text{place}}) \leftarrow \text{FirstAction}(n_{\text{best}})\)
10. **return** \((T_{\text{pick}}, T_{\text{place}})\)

V. EXPERIMENTS

In this section, we answer three questions: (1) Is TVF able to generalize to unseen tasks in a zero-shot manner after training on a handful of expert demonstrations? (2) Does TVF work on real robot platforms? (3) Does the proposed VF model produce better prediction results than baseline methods? Ablation studies and more experiment details can be found in the supplementary materials on the project website.

A. Simulation Experiments

Fig. 1(a) shows our simulation experiment setup which builds on an open-source manipulation task simulation environment Ravens [10]. A UR5 robot arm with a suction gripper is used to perform pick-and-place actions in a 0.5 \(\times\) 0.5 m² tabletop workspace. Three RGB-D cameras are used to reconstruct the top-down observation \(o_t\) of the workspace. We design 14 different block rearrangement tasks (Fig. 5). In each task, the blocks are randomly positioned and oriented within the workspace. All tasks require multiple steps to finish. All tasks are multi-modal – there may be multiple valid actions to perform in a step. A scripted oracle is written for each task to provide expert demonstrations leading to the goal configuration.

We divide the 14 tasks into 6 training tasks and 8 unseen tasks which are not present in the training data (Fig. 5). For each training task, we collect 1000 expert demonstrations. When testing the performance on training with different numbers of demonstrations (Sec. V-D), we sample from these demonstrations as training data. To make theVF model more flexible, two random actions, which pick a block on the tabletop and place it at a collision-free pose, are also included in the collection of each expert demonstration. Both random actions and oracle actions are used for training theVF model; only oracle actions are used for training the GCTN for the multi-modal action proposal. For all the unseen tasks, we collect 20 demonstrations for testing.

We compare our method with GCTN [9], a state-of-the-art method for learning robot rearrangement tasks. We also compare different variants of TVF: we vary the number of clusters \(K\) in K-Means Clustering and the maximum depth \(d_{\text{max}}\) of the task. The cluster numbers \(K\) for TVF-Small and -Large are 2 and 3, respectively. The tree depths \(d_{\text{max}}\) are 1 and 3, respectively. Each step takes about 0.08s, 0.14s, and 1.82s for GCTN, TVF-Small, and TVF-Large on an NVIDIA RTX 3090 GPU, respectively. We evaluate the performance of a method with the success rate. The maximum step number for a rollout equals the number of blocks in the task. A trial is considered successful if the planar translation, the z-coordinate, and the rotation about the z-axis of all the blocks to the corresponding target poses are less than 1cm, 0.5cm, and 15°, respectively. We train GCTN and TVF variants with 1, 10, 100, and 1000 demonstrations per training task (6, 60, 600, 6000 demonstrations in total).

Following [9], we use different TensorFlow seeds to train 3 models for all methods. We test each model on the test data and report the average result of the 3 models.

B. Real Robot Experiments

Fig. 1(b) shows the setup of our real robot experiments. We implement our method on a Franka Emika Panda robot arm with a suction gripper. A PrimeSense Carmine 1.09 RGB-D camera is mounted on the end-effector of the robot. As in the simulation, the workspace is a 0.5 \(\times\) 0.5 m² tabletop. At the beginning of each task, the goal block configuration is shown to the robot and the blocks are then disassembled and placed randomly on the table. At the beginning of each step, the robot goes to a predefined configuration to capture the top-down observation of the current scene. We use the height from the observation of the picking and placing actions. IKFast [36] and MoveIt [37] are used for motion planning. We pre-process the RGB images and heightmaps by filtering out the background. We found that GCTN is not able to learn
TABLE I: Average Success Rates of Simulation Experiments on Unseen Tasks. We show the average success rate (%) on the test data of unseen tasks v.s. # of demonstrations (1, 10, 100, or 1000) per task in the training data. Higher is better.

| Method          | 1     | 10    | 100   | 1000  |
|-----------------|-------|-------|-------|-------|
| GCTN            | 1.3   | 55.4  | 49.0  | 54.2  |
| TVF-Small (K = 2, \(d_{\text{max}} = 1\)) | 1.7   | 71.5  | 62.3  | 72.5  |
| TVF-Large (K = 3, \(d_{\text{max}} = 3\)) | 2.9   | 78.5  | 71.7  | 85.6  |

well without the background filtering. More details can be found in the supplementary materials on the project website.

C. Training Details

We implement our method with TensorFlow. Taking advantage of the spatial consistency of our method (Sec. IV-A), we apply extensive data augmentation of random translations and rotations to the training data to train the VF model. We use the L1 loss and the Adam optimizer with a learning rate of \(1 \times 10^{-4}\) and train for 60000 iterations. For the L1 loss, the weight for the height channel is five times larger than those for the RGB channels. To train GCTN, we use exactly the same training setting as [9]. Both simulation and real robot experiments are trained with the same setting as described above.

D. Result I: Simulation Experiments

We evaluate on zero-shot generalization to unseen tasks which are not present in the training data. We use the models trained with the data of the 6 training tasks and test on the 8 unseen tasks in Fig. 5 without additional training. The average success rates are shown in Tab. I. Both TVF variants outperform GCTN on unseen tasks in all cases of demo numbers. When there are more than 1 demo for each training task, TVF variants outperform GCTN by a large margin. Remarkably, TVF-Large achieves an average success rate of 78.5% when only 10 demos per task is given for training, while vanilla GCTN achieves 55.4%. In the case of 1000 demos per task, TVF-Large achieves a remarkable average success rate of 85.6%. For the TVF variants, the performance improves with the increase of the complexity of the tree. The success rates of GCTN and TVF variants are very low when there is only 1 demo per training task and increase by a large margin when given 10 demos per task. In general, the performance of GCTN and TVF variants increases with the increase of demo numbers. However, this is not always the case: the performance of 100 demos is worse than that of 10 demos. The original GCTN paper reports similar results [9].

The success rates for each unseen task are shown in Tab. II. In the cases of 10, 100, and 1000 demos, both TVF variants outperform GCTN in every task. For Building with 1000 demos, GCTN achieves an average success rate of 3.3% while TVF-Large improves the performance up to 25.0%. Both GCTN and TVF variants are able to achieve good results on simple tasks (e.g., Plane Square and Plane T). But for more complicated tasks, GCTN struggles while TVF variants perform much better. Another interesting observation is that the performance of TVF variants correlates with that of GCTN. This is because the action proposal module is based on GCTN – if GCTN is not good, the multi-modal action proposal will not be good either.

E. Result II: Real Robot Experiments

As we are able to train TVF with only a handful of expert data, this makes real robot experiments possible. We collect 30 expert demonstrations for 3 training tasks (10 demos per task) and use these data to train TVF and GCTN. We test TVF and GCTN on both the 3 training tasks and 3 unseen tasks. Each task is tested with 10 rollouts. The results are shown in Tab. III and Fig. 6. Our method works in the real world. And it outperforms GCTN in 5 of the 6 tasks and is on par with GCTN on Rectangle. Notably, while GCTN fails in all 10 rollouts in the unseen Twin Tower task, our method is able to achieve a success rate of 60%.

F. Result III: Visual Foresight Model

Finally, we evaluate our VF model on predicting the next-step observation \(o_{t+1}\) from the current observation \(o_t\) and action \(a_t\) given a small number of training data. We compare with a baseline method, referred to as Latent Dynamics, which instead encodes the action in the
Combining the VF model and the action proposal module with a tree-search algorithm, we propose a typical reason for the failures of GCTN is the single-modal action proposal. In both training tasks and unseen tasks in the simulation (60 demos in total). For both methods, we use different TensorFlow seeds to train 3 models and report the average result of the 3 models on test data in Tab. IV. Our VF model outperforms Latent Dynamics in both training and unseen tasks.

VI. DISCUSSIONS & FUTURE WORK

A typical reason for the failures of GCTN is the single-modal action proposal. In both training tasks and unseen tasks in real robot experiments (Fig. 6(a)), we observe that the action with the highest value of GCTN is sometimes incorrect. And since GCTN is single-modal, it will take the incorrect action and fail the task. On the other hand, even if the action with the highest value is incorrect, TVF is able to predict which actions in the multiple proposals will better lead to the goal with the VF model and take the action. This brings about substantial advantages on unseen tasks compared to GCTN. See the columns highlighted with a * in Tab. II and III. Our VF model features efficient training which is able to predict accurate next-step observations given only tens of training data. This is achieved by introducing inductive biases in the network design – encoding the action in the image space in a spatially consistent way allows our VF model to take advantage of the translational equivariance property of the FCN. The performance advantage over Latent Dynamics in Tab. IV justifies our design choice.

As TVF assumes no prior knowledge of objects, we envision it can be generalized to more variable objects. Moreover, although we focus on $SE(2)$ tabletop rearrangement in this paper and use FCNs as the network architecture, future work can explore applying a more advanced architecture [38] that preserves $SE(2)$ equivariance to further improve sample efficiency. Another possible direction is to extend the idea of geometry-aware visual foresight to more general settings (e.g., 3D workspaces described by point cloud data [39]) and develop VF models for accomplishing more challenging tasks. In addition, our VF model is deterministic, i.e., there is no stochasticity to address the uncertainty of robot actions. Future work can also explore using uncertainty-aware models (e.g., Bayesian Neural Network [40]) to model the visual dynamics. Finally, the number of clusters $K$ is fixed for a particular TF variant is fixed and equals $K$. Future work can investigate methods to adaptively generate different numbers of actions according to the action-value maps $Q_{\text{pick}}$ and $Q_{\text{place}}$.

VII. CONCLUSIONS

In this paper, we propose a simple visual foresight (VF) model which is able to predict the next-step observation from the current observation and a pick-and-place action. The VF model is able to learn efficiently from only a handful of training data. In addition, we propose a multi-modal action proposal module which builds on a state-of-the-art imitation learning method [9] for more versatile action proposal. Combining the VF model and the action proposal module with a tree-search algorithm, we propose...
Transporters with Visual Foresight (TVF), a novel method for rearrangement task planning from image data which is able to achieve zero-shot generalization to unseen tasks with only tens of expert demonstrations. Results show that our method outperforms a state-of-the-art baseline method on average success rates of unseen tasks in both simulation and real robot experiments. Our proposed VF model outperforms a baseline method when only given a small number of training data. Robotic systems that can generalize their function to scenarios beyond interpolations of those that are previously seen represents a higher level of capability.

REFERENCES

[1] M. E. Seligman, P. Raitlor, R. F. Baumeister, and C. Sripada, “Navigating into the future or driven by the past,” Perspectives on psychological science, vol. 8, no. 2, pp. 119–141, 2013.

[2] 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.

[3] C. Paxton, Y. Barony, K. Katyal, R. Arora, and G. D. Hager, “Visual robot task planning,” in 2019 international conference on robotics and automation (ICRA). IEEE, 2019, pp. 832–838.

[4] 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,” arXiv preprint arXiv:2003.09044, 2020.

[5] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, “Neural task programming: Learning to generalize across hierarchical tasks,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 3795–3802.

[6] B. Huang, S. D. Han, A. Boularias, and J. Yu, “ Dip: Deep interaction prediction network with application to clutter removal,” in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 4694–4701.

[7] B. Huang, S. D. Han, J. Yu, and A. Boularias, “Visual foresight tree for object retrieval from clutter with nonprehensile rearrangement,” arXiv preprint arXiv:2105.02857, 2021.

[8] H. Suh and R. Tedrake, “The surprising effectiveness of linear models for visual foresight in object pile manipulation,” arXiv preprint arXiv:2002.09093, 2020.

[9] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, “Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks,” in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 4568–4575.

[10] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani et al., “Transporter networks: Rearranging the visual world for robotic manipulation,” arXiv preprint arXiv:2010.14406, 2020.

[11] M. H. Lim, A. Zeng, B. Ichter, M. Bandari, E. Coumans, C. Tomlin, S. Schaal, and A. Faust, “Multi-task learning with sequence-conditioned transporter networks,” arXiv preprint arXiv:2109.07578, 2021.

[12] M. Shridhar, L. Manuelli, and D. Fox, “Cliport: What and where pathways for robotic manipulation,” in Conference on Robot Learning. PMLR, 2022, pp. 894–906.

[13] H. Huang, D. Wang, R. Walter, and R. Platt, “Equivarient transporter network,” arXiv preprint arXiv:2202.09400, 2022.

[14] F. Ebert, C. Finn, A. X. Lee, and S. Levine, “Self-supervised visual planning with temporal skip connections,” in CoRL, 2017, pp. 344–356.

[15] J. Kossen, K. Stelzner, M. Hussing, C. Voelcker, and K. Kersting, “Structured object-aware physics prediction for video modeling and planning,” arXiv preprint arXiv:1910.02425, 2019.

[16] M. Minderer, C. Sun, R. Villegas, F. Cole, K. Murphy, and H. Lee, “Unsupervised learning of object structure and dynamics from videos,” arXiv preprint arXiv:1906.07889, 2019.

[17] M. Toussaint, “Logic-geometric programming: An optimization-based approach to combined task and motion planning,” in Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.