Graph-Structured Visual Imitation

Maximilian Sieb*, Zhou Xian*, Audrey Huang, Oliver Kroemer, Katerina Fragkiadaki
Carnegie Mellon University
`{msieb, xianz1, audreyh, okroemer, kfragki2}@andrew.cmu.edu`

Abstract: We cast visual imitation as a visual correspondence problem. Our robotic agent is rewarded when its actions result in better matching of relative spatial configurations for corresponding visual entities detected in its workspace and the teacher's demonstration. We build upon recent advances in Computer Vision, such as human finger keypoint detectors, object detectors trained on-the-fly with synthetic augmentations, and point detectors supervised by viewpoint changes [1] and learn multiple visual entity detectors for each demonstration without human annotations or robot interactions. We empirically show that the proposed factorized visual representations of entities and their spatial arrangements drive successful imitation of a variety of manipulation skills within minutes, using a single demonstration and without any environment instrumentation. It is robust to background clutter and can effectively generalize across environment variations between demonstrator and imitator, greatly outperforming unstructured non-factorized full-frame CNN encodings of previous works [2].

Keywords: Imitation learning, Robotic manipulation, Reinforcement learning

1 Introduction

Humans learn skills by watching other humans [3]. The ability to learn from observation —called visual imitation [4] or third-person imitation [5]— has always been a much-desired goal in artificial intelligence as a means of quickly programming agents in an intuitive manner, as opposed to hard-coding their behaviors. Visual imitation requires fine-grained understanding of the demonstrator's visual scene and its changes over time. The imitator then will use its own embodiment and dynamics to cause a similar change in its own environment. Visual imitation then boils down to learning a visual similarity function between the demonstrator's and imitator's environments, whose maximization—via the imitator's actions—would result in correct skill imitation. This similarity function determines which aspects of the visual observations are relevant to reproducing the demonstrated skill, i.e., it defines what to imitate and what to ignore [6].

We propose hierarchical graph video representations, called Visual Entity Graphs (VEGs), where nodes represent visual entities (objects, parts or points) tracked in space and time, and edges represent their relative 3D spatial arrangements. Our video graph encoding is based on the observation that the appearance of the scene in some level of abstraction (objects, parts or points) remains constant over time, but the spatial arrangements of entities change over time, an observation which follows directly from the laws of Newtonian physics. The proposed hierarchical visual entity graphs disentangle the "what" and "where" of the scene in multiple levels of abstraction: nodes represent visual entities that persist over time, and edges within each graph represent their relative 3D spatial arrangements that may change over time. For each pair of timesteps, we build two VEGs, one for the demonstrator and one for the imitator. Their nodes are in one-to-one correspondence, as shown in Figure 1. Our imitation reward function then measures agreement of the relative spatial configurations between corresponding node pairs, and guides reinforcement learning of manipulation tasks from a single video demonstration using a handful of real-world interactions.

Under the proposed VEG encoding, visual imitation boils down to learning to detect corresponding visual entities (objects, parts, or points) between the demonstrator's and imitator's environments.

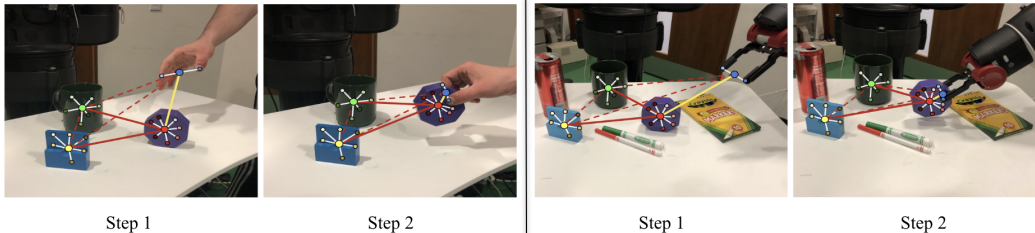|			
Step 1	Step 2	Step 1	Step 2

Figure 1: **Graph-structured visual imitation.** We show the VEGs for a human demonstration and robot imitation for two timesteps. Corresponding nodes in the human demonstration (left) and robot imitation (right) share the same color. The graphs are hierarchical. Edges exist between object, robot and human hand nodes and point feature nodes, and are added and deleted dynamically over time based on motion saliency, as shown in the figure with solid and dashed lines, respectively. Our graph representation is robust to viewpoint variation between demonstrator and executor, and can handle cluttered backgrounds, as illustrated in the figure.

This requires fine-grained visual understanding of both demonstrator's and imitator's environments. The challenge in this visual parsing problem is that objects used by the demonstrator are often not included in the labelled image or object categories of ImageNet [7] and MS-COCO [8] datasets, making off-the-shelf pre-trained object-detectors less useful. We instead opt for scene-specific self-supervised detectors for points and objects. We use self-supervised point visual feature detectors trained by viewpoint changes, and visual detectors of objects and parts trained from synthetically generated images that augment a video at hand. Lastly, we also use human hand keypoint detectors to parse the demonstrator's hand trajectories. The proposed scene-conditioned visual entity detectors establish correspondences between the demonstrator's and imitator's workspaces, despite differences in occlusion patterns, viewpoint changes or robot-human body visual discrepancies (Figure 1). We use the resulting reward function for trajectory optimization [9], and show that it can imitate a single human demonstration from a handful of real world trials on a Baxter robot.

In summary, our contributions are as follows: i) We propose a what-where hierarchical graph visual encoding for visual correspondence estimation. ii) We propose scene-specific point and object visual detectors, as well as human hand detectors. To the best of our knowledge, this is the first work that uses human finger visual detectors as opposed to environment instrumentation [10] to track the demonstrator's hand for visual imitation learning of object manipulations. iii) We imitate using a single demonstration, without any robot random exploration as in [11, 4], or any data of the robot performing the task as in [2]. We do so without ever having access to expert actions. iv) We show imitation results on a real robotic platform.

We compare our proposed representation against full-frame image encodings of previous works [2, 11] that do not use a what-where decomposition during matching. Our experiments suggest they require a very large number of video examples of humans and robots executing the task to acquire generalization abilities similar to our method. They fail to imitate the demonstrated skill most of the times, as we show in our experiments. When humans imitate fellow humans, they are equipped with excellent visual detectors, visual feature extractors, and motion estimators as opposed to learning those from scratch for every new task. We opt for a similar transfer of machine vision knowledge during imitation for robotic agents.

Our code and accompanying videos are included in the supplementary materials.

2 Related Work

Visual imitation learning. Imitation learning addresses the problem of learning skills by observing expert demonstrations [6]. However, most previous approaches assume that expert demonstrations are given in the workspace of the agent, (e.g., through kinesthetic teaching or teleoperation [12, 13]) and the actions/decisions of the expert can be imitated directly [14]. Imitating humans based on visual information is much more challenging due to the difficult visual inference needed for fine-grained activity understanding [5]. In this case, a mapping between observations in the demonstrator space to observations in the imitator space is required and is essential for successful

imitation [15]. Numerous works bypass the difficult perception problem by using special instrumentation of the environment, such as AR tags, to read off object and hand 3D locations and poses during video demonstrations, and use rewards based on known 3D object goal configurations [16, 17]. Other works use hand-designed reward detectors that work only in restrictive scenarios [18]. Direct matching of pixel intensities is not a meaningful measure of similarity [19] as it is easily spoiled by difference in viewpoints, human body versus robot body parts, illumination changes, or changes in object poses.

Recent approaches to attempt instead to *learn* such visual similarity by training and matching whole image feature embeddings directly, and avoid explicit extraction of the scene structure in terms of objects and their 3D poses. Numerous objectives have been proposed to learn full image or image sequence convolutional image embeddings, such as multiview invariant and time-contrastive objectives in [2, 20], forward and inverse dynamics model learning in [4, 21], or reconstruction and temporal prediction objectives in [22, 23]. Work of [24] provides an overview of common objectives and inductive biases for state representation learning. However, the data used to train such image embeddings are both human video demonstrations as well as *robot executions of the task*, or parts of the task, so that the neural network embedding function learns to be robust to the presence of the robotic gripper or human hand. Yet, the requirement of the robot executing the task beats the purpose of visual imitation, and brings it closer to kinesthetic teaching. The same holds for recent work of [25], which learns an image encoding via frame prediction using robot's execution data, albeit the paper title. Instead, learning our graph video encoding does not require robot executions.

Similar to our work, work of [26] also uses 3D spatial object arrangements to guide visual imitation of manipulation tasks. However, they do not consider human keypoints or any entities finer than objects, which suggests their method can only imitate simple translation tasks, where object pose is not relevant (e.g., they cannot handle rotation). Work of [27] utilizes human pose detectors to imitate 3D human motion extracted from YouTube videos of acrobatic activities in simulation. However, no contact with objects is considered, and imitation of human motion only happens in a simulated agent. In comparison, we do not imitate motion alone, but rather, we carry out a desired manipulation of the environment. Human hands are part of the graph we attempt to create, but so are the surrounding objects in the scene.

Scene graphs, object-centric reinforcement learning, and relational neural networks. Representing a visual observation in terms of objects or parts and their pairwise relations has been found beneficial for generalization of action-conditioned scene dynamics [28, 29, 30], body motion and person trajectory forecasting [31, 32], and reinforcement learning [33]. Such graph-encodings have also been used to learn a model of the agent [34], and use it for model-predictive control in non-visual domains. Work of Devin et al. [35] uses pretrained object detectors and learns attention over the obtained detection boxes, which are incorporated as part of the state representation for policy learning. The graph representation we propose in this work not only employs explicit attention to relevant objects, parts, and points, but also preserves their correspondence in time, i.e., the detectors *bind* with specific objects, parts, points over time.

3   Formulation

We encode a demonstration video of length T provided by the human expert and an imitation video of the same length T provided by the imitator in terms of two graph sequences $\mathcal{G}_D^t = \{\mathcal{V}_D^t, \mathcal{E}_D^t\}, t = 1 \cdots T$ and $\mathcal{G}_I^t = \{\mathcal{V}_I^t, \mathcal{E}_I^t\}, t = 1 \cdots T$, respectively. We omit the subscript D or I when either it is clear from the context or it is not important to which workspace we are referring to. A node $\mathcal{V}_i^t$ corresponds to the ith visual entity and its respective 3D (X, Y, Z) world coordinate $\mathbf{x}_i^t$, and an edge $\mathcal{E}_{(i,j)}^t$ correspond to a 3D spatial relation between two node entities to be preserved during imitation. We define a visual entity node $\mathcal{V}_i^t$ to be any object, object part, or point that can be reliably detected in the demonstrator's **and** imitator's workspace. All nodes are in one-to-one correspondence between the demonstrator and imitator graph, as shown in Figure 1. An entity can dynamically appear and disappear over time. We only require each entity associated with the demonstration sequence to have a corresponding entity in the imitation sequence.

We consider three types of nodes: object nodes $\mathcal{V}_o$, point nodes $\mathcal{V}_p$, and hand/robot nodes $\mathcal{V}_h$. An object node represent any rigid or non-rigid object that constitutes a separate physical entity in the world, while a point node represents any 3D physical point on an object, as seen in Figure 1. Hand
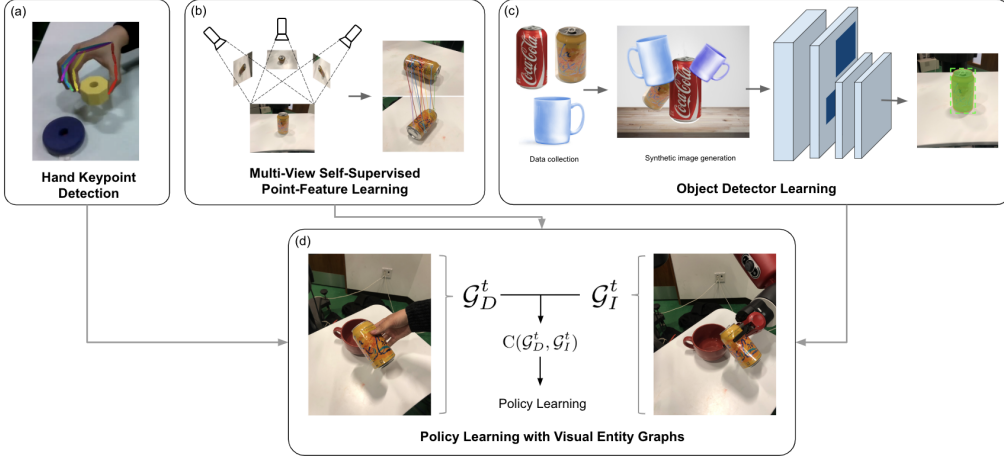
Figure 2: **Detecting visual entities.** We use human hand keypoint detectors, multi-viewpoint feature learning, and synthetic image generation for on-the-fly object detector training from only a few object mask examples. Using a manually designed mapping between the human hand and the robot, the visual entity detectors can effectively bridge the visual gap between demonstrator and imitator environment, are robust to background clutter, and generalize across different object instances.

and robot nodes represent the human wrist 3D location, and the robotic gripper center 3D location, respectively. We do not consider edges between point nodes. Rather, each point node is connected only to the object node it is part of. In that sense, our graphs are hierarchical.

Our cost function at each time step t measures visual dissimilarity across the demonstrator and imitator graphs $\mathcal{G}_D^t$ and $\mathcal{G}_I^t$ in terms of relative spatial arrangements of corresponding entity pairs, as follows:

$$\mathcal{C}(\mathcal{G}_D^t, \mathcal{G}_I^t) = \sum_{i,j,i<j} w(\mathcal{E}_{(i,j)}^t) \cdot \text{att}(\mathcal{E}_{(i,j)}^t) \cdot \left\| \left(\mathbf{x}_{D,i}^t - \mathbf{x}_{D,j}^t \right) - \left(\mathbf{x}_{I,i}^t - \mathbf{x}_{I,j}^t \right) \right\|, \tag{1}$$

where $\text{att}(\mathcal{E}_{(i,j)}^t) \in \{0,1\}$ is a binary attention function that determines whether a particular edge is present depending on the motion of the corresponding nodes, and $w(\mathcal{E}_{(i,j)}^t) \in \mathbb{R}$ denotes edge weights. We tie weights across all edges of the same type, namely, object-hand edges, object-object edges and object-point edges. They are hyper-parameters of our framework and we set them them empirically. Learning to adjust those weights per task is an interesting and straightforward direction which wee leave for future work.

3.1 Detecting Visual Entities

We define a visual entity to be any object, object part, or point that can be reliably detected in the demonstrator's and imitator's workspace. For imitating fine-grained manipulation of an object, inferring the translation of its bounding box is not enough, rather, the object's 3D pose and deformation needs to be inferred and imitated. A central design choice in our work is using point feature detectors and motion of the detected points to infer the object's change of pose between demonstrator's and imitator's environments, without additional learning, as opposed to training object appearance features extracted within the object's bounding box to encode object change pose or deformation. We opt for what-where decomposition of the object's appearance, as opposed to whole object box embedding learning.

We assume no access to human annotations that would mark relevant corresponding entities across demonstrator's and imitator's environments. We instead train scene-specific object and point detectors for entities that can be reliably recognized across demonstrator's and imitator's workspaces, and human hand keypoint detectors for tracking the human hand. The point detector re-samples points at each step randomly on detected area of objects in demonstration and computes corresponding points in the imitator's view, and is thus robust to partial occlusions. In case of full occlusion, our hand and

object detectors use last known location in the past. Thus, our detection pipeline possesses certain robustness to object occlusions and possible detector failures.

Human hand keypoint detectors. We make use of state-of-the-art hand detectors of Simon et al. [36] to detect human finger joints, and obtain their 3D locations using a D435 Intel RealSense RGB-D camera. We rely on forward kinematics and a calibrated camera with respect to the robots coordinate frame to detect the 3D locations of the tips of the robots end-effector. We map the finger tips of a Baxter robot's parallel-jaw gripper to the demonstrator's thumb and index finger tips. We detect grasp and release actions by thresholding the distance between the two finger tips of the human during the demonstration of the task. End-to-end approaches such as [37] rely on large amounts of data to train hand-to-robot correspondences, and are therefore prohibitive in few-shot learning scenarios, whereas our method works thanks to the thousands of labelled hand examples the human hand detectors of [36] has seen.

Point feature detectors from cross-view correspondence. An agent that has access to its egomotion and observes a static scene from multiple views can infer visual correspondences across views through triangulation [38]. We use these self-generated visual correspondences to drive visual metric learning of deep feature descriptors that are robust to changes in the object pose or camera viewpoint. After training, we match such point features across imitator's and demonstrator's environments to establish correspondence [1]. We collect multiview image sequences of the workspace of the robotic agent in an automatic fashion: we use an RGB-D camera attached to the robot's end-effector and move the camera while following random trajectories that cover many viewpoints of the scene and at various distances from the objects. We use the robot's forward kinematics model to estimate the camera poses via hand-eye calibration, which, in combination with the known intrinsic parameters and aligned depth images, allows for robust 3D-reconstruction of the scene and provides accurate pixel correspondences across different viewpoints. The complete feature learning setup is illustrated in Figure 2(b). During training, we randomly sample image pairs and generate a number of matching and non-matching pixel pairs. We then minimize pixel-wise contrastive loss [39, 40], which forces matching pixels to be close in the learned feature spaces, while maintaining a distance margin for non-matching pixels. This point feature learning pipeline produces a mapping from an RGB image to dense per-point descriptors. Even though supervision comes via within-instance correspondences, generalization across different objects is expected due to the limited capacity of the network model. This enables our VEG representation with powerful generalizability to novel objects unseen in demonstration, as shown in Figure 3b. We use ResNet-34 as our backbone, and learn a 4-dimensional point embedding vector for each pixel in the image.

Synthetic data augmentation. We use background subtraction to propose object 2D segmentation masks, and train a visual detector for each mask using synthetic data augmentation, as shown in Figure 2. Specifically, we create a large synthetic dataset by translating, scaling, rotating and changing the pixel intensity of the extracted RGB segmentation masks. The object masks often partially overlap with one another in the synthetic images. These overlaps help the agent learn to detect amodal object boxes under partial occlusions. Since we generate such images, we automatically know the groundtruth bounding box and mask that correspond to each object in each image. We then finetune a Mask R-CNN object detector [41]—initialized from weights learned under the object detection and segmentation task in MS-COCO—to predict boxes and masks for the synthetically generated images.

3.2 Motion Saliency for Dynamic Graph Construction

We use similar motion saliency heuristics to decide dynamically over time what edges (i, j) to consider in our imitation cost function of Eq. 1, by setting $\text{att}(\mathcal{E}^{t,(i,j)}) = 1$ to denote edge presence. We define an anchor object to be any object in motion, and in the case of no moving objects—e.g., when the demonstrator is simply reaching towards an object—we define the anchor object to be the closest-in-the-future moving object. We consider edges between the anchor object node and all other *corresponding* object nodes in the scene, as well as the hand/robot node, and all point nodes that belong to the anchor object. This type of motion attention is a well-established principle that drives human attention when imitating other humans. Work of [42] uses AR markers to estimate such task relevance based on motion attention. We simply use our object detection network to track objects in time to infer movement between the different objects in the scene.

3.3 Policy Learning with Visual Entity Graphs

Our goal is the robot to imitate the intended object manipulation task from a single human visual demonstration. We formulate this as a reinforcement learning problem, where at each time step
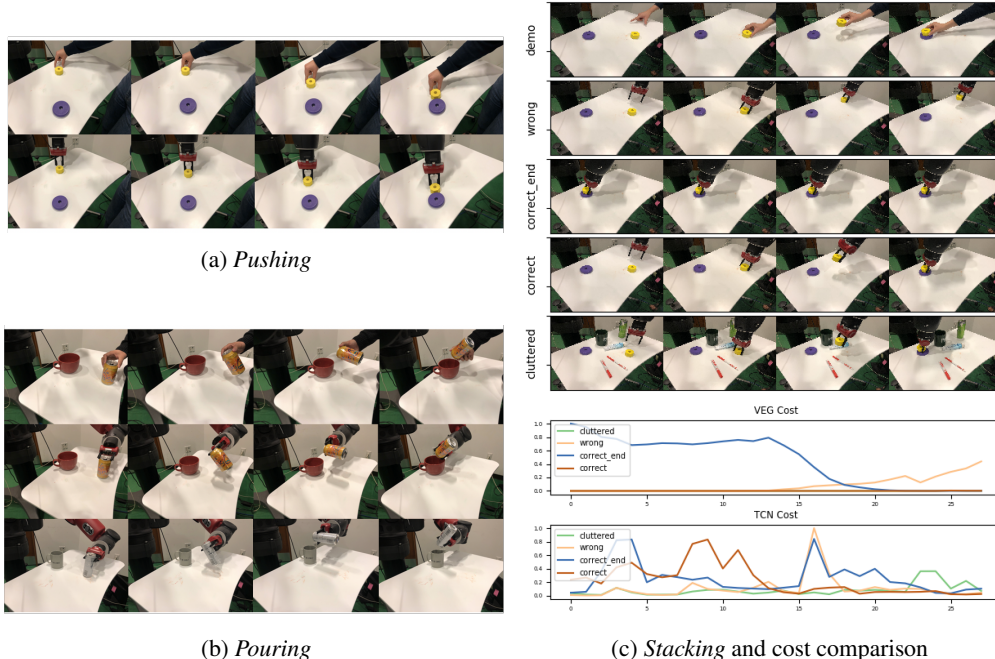
(a) *Pushing*

(b) *Pouring*

(c) *Stacking* and cost comparison

Figure 3: Examples of the imitation tasks we consider. (a): *pushing*, (b): *pouring* (c): *stacking* and **cost comparison of the proposed VEG cost function and the TCN embedding cost [2].** The VEG based cost curves are highly discriminative of the imitation quality for the stacking task, in contrast to TCN. Costs are scaled between $[0, 1]$ by dividing by the maximum absolute value of the respective feature space for visualization purposes. `correct_end` is simply repeating the last frame of a successful imitation, showing our lost function serves well as an attractor towards the final goal configuration. Note that `correct` and `cluttered` overlap due to both having zero VEG cost, showing that our method is robust to background clutter.

the cost is given by Eq. 1. We use PILQR [9] to minimize the cost function in Eq. 1, a state-of-the-art trajectory optimization method that combines a model-based linear quadratic regulator with path integral policy search to better handle non-linear dynamics. We learn a time-dependent policy $\pi_t(\mathbf{u}_t|\mathbf{x}_t; \theta) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{\Sigma}_t)$, where the time-dependent control gains are learned by alternating model-based and model-free updates, where the dynamical model $p(\mathbf{x}_t|, \mathbf{x}_{t-1}, \mathbf{u}_t)$ of the *a priori* unknown dynamics is learned during training time. The actions $\mathbf{u}$ are defined as the changes in the robot end-effector's 3D position and orientation about the vertical axis, giving a 4-dimensional action space. The state representation—over which we learn linear dynamics— consists of the joint angles, end-effector position, and the graph configuration of the scene, concatenated into one vector. For N objects, this featurization scheme results in a $3 + n_{joints} + \sum_{i=1}^{N_{anchors}}(N_{total} - 1) * 3 + \sum_{i=1}^{N_{anchors}} dim(\phi)$ dimensional state space, where n_{joints} denotes the number of joints of the robot and ϕ encodes the chosen node feature for a visual entity in the graph. In our case, we perform uniform sampling of node-specific point features at each time step, and we directly incorporate the averaged pairwise distance of all pixels across demonstration and imitation into the state space, yielding $dim(\phi) = 1$. We further use behavioral cloning to infer opening and closing of the robot gripper by thresholding the distance between the human index finger and thumb during the demonstration.

4   Experiments

We test our visual imitation framework on a Baxter robot. We consider the following tasks to imitate: i) **Pushing**: The robot needs to push an object while following specific trajectories. ii) **Stacking**: The robot needs to pick up an object, put it on top of another one and release the object. iii) **Pouring**: The robot needs to pour liquid from one container into another.

6

For every task, we train corresponding object detectors using synthetic data augmentation and point features using multi-view self-supervised feature learning. Note that both processes are fully automated and **do not require human demonstrations or robot interactions**. Rather, it suffices to use a camera that is setup to move around the scene in a prerecorded fashion

We compare our method against time-contrastive networks (TCN) of Sermanet et al. [2]. We are not aware of other instrumentation-free methods that have attempted single shot imitation of manipulation skills, without assuming extensive pre-training with random actions for model learning [11]. TCN trains an image embedding network using a triplet ranking loss [43], ensuring that temporally near pairs of frames are closer to one another in embedding space than any temporally distant pairs of frames. In this way, the feature extractor focuses on the *dynamic* part of the demonstration video. We implemented the TCN baseline using the same architecture as in [2], which consists of an Inception-v3 model [44], pretrained on ImageNet, up to the Mixed 5d layer which is then followed by two convolutional layers, a spatial softmax layer and a fully-connected layer. The network finally outputs a 32-dimensional embedding vector for the input image. For each imitation task, we train a corresponding TCN using 10 video sequences, 5 human demonstrations and 5 robot executions while performing tasks with relevant objects and environment configuration, together with the human demonstration we provide for learning the policy with VEG. With respect to the policy learning with TCN, we use a cost function of the form $\alpha||\mathbf{z}_t - \mathbf{w}_t||_2^2 + \beta\sqrt{\gamma + ||\mathbf{z}_t - \mathbf{w}_t||_2^2}$, where we choose $\gamma = 10^{-5}$, $\alpha = 1.0$, and $\beta = 0.001$. Here, $\mathbf{z}_t$ and $\mathbf{w}_t$ denote the state embedding at each time step for the imitation and demonstration, respectively. While our method uses only a single human demonstration, it does require training object detection and point-feature networks, although both the data synthesizing for the object detector and the data collection for the point-feature network are fully automated. To ensure a fair comparison, we put the same amount of time effort into data collection for training the TCN baseline.

Our experiments aim to compare the proposed graph structured encoding against convolutional image encodings of previous work [2] for imitating skills, evaluate the robustness of our method against detectors' failures and occlusions, evaluate its robustness to variability in the objects' spatial configurations & background clutter and its generalizability across objects with different shapes and textures.

Reward Shaping. In Figure 3c, we show reward curves for our method and the TCN baseline for each robot execution, measuring how well the robot is imitating the human demonstration. The horizontal axis denotes time and the vertical axis denotes imitation cost. The proposed graph-based cost function correctly identifies all correct robot imitations, despite heavy background clutter in the 5th row, and correctly signals the wrong imitation segments in 2nd and 3rd rows. In contrast, the baseline TCN cost curves are non-discriminative. Highly discriminative cost curves are critical for effective policy learning, which we discuss right below.

Task - Pushing. In this task, the robot needs to push the yellow octagon towards the purple ring following the trajectory showed by the demonstrator (Figure 3a). We evaluate three task variations: i) *straight-line:* Pushing the object following a straight line. ii) *straight-line-grasped:* Moving the yellow octagon along a straight line while it is being grasped. iii) *direction-change:* Pushing the yellow octagon along a trajectory with a sharp direction change of 90 degrees. Imitating such direction change requires the robot to change the point of contact with the object during pushing. For straight-line pushing, we attempted imitation in two different environments, one with a yellow octagon block, and one with a smaller orange square block to evaluate the generalization capability of our graph-based framework across objects with different geometries. Essentially, we substitute the object detector for the small orange square block in the second case for the detector of the yellow octagon block.

Task	object	VEG	TCN
straight-line	yellow octagon	5/5	1/5
	orange square	4/5	0/5
straight-line-grasped	yellow octagon	Success	Failure
direction-change	yellow octagon	Success	Failure

(a)

Task	object	VEG	TCN
stacking	yellow octagon	4/5	0/5
	orange square	3/5	0/5
simple-stacking	yellow octagon	5/5	3/5

(b)

Table 1: Success rates for the pushing and stacking tasks.

In order to test the robustness of VEG to variations in the objects' spatial configurations, for the basic straight-line pushing task, we randomly perturb the starting configurations of all the objects and the robot's end-effector within a norm ball of diameter 6cm, and run the policy training 5 times for each object. We consider the task solved if the robot is able to push the object within 1cm of the desired target position. We report the success rate for each case in Table 1(a). The robot successfully solves the task for all 5 runs for the original yellow octagon, and for 4 runs out of 5 for the smaller orange square, demonstrating robustness against perturbation in objects' spatial configuration and strong generalization over objects with novel geometries. TCN failed in almost all runs, and in case for simple straight-line pushing, it failed to push the object into the target region even with significantly more iterations. For the straight-line-grasped task, the robot is forced to keep the gripper closed. Hence, the robot cannot simply hard-imitate the hand trajectory, but rather needs to follow a trajectory that differs from the human demonstrator to successfully solve the task. In this sense, the hand trajectory of the human serves as a weak guiding signal during the policy learning. The robot solves both straight-line-grasped and direction-change pushing after 8 iterations of trajectory optimization. Note that we increased the hand edge weight w to account for stronger guidance of the hand during the direction-change task. TCN fails to solve any of the latter two tasks, generally performing poorly in tasks where the robot does not have continuous contact with the object.

Task - Stacking. In this task, the robot needs to reach and grasp the yellow octagon and position it on top of the purple ring, as shown in the 1st and 4th rows in Figure 3c. As in our pushing experiments, we randomize the starting configurations of the scene and the robot, and evaluate our method on this augmented setup for stacking both yellow octagon and the orange square. We report the results in Table 1(b). VEGs enables robust policy learning for the yellow octagon, and shows generalization towards objects with novel geometries that have not been observed during demonstration. Being flatter than the octagon, the orange square is much harder to grasp, yet the robot is able to learn the skill successfully most of the times. The TCN baseline is not able to learn a successful grasping action, which requires precise and coordinated end-effector movement. We then evaluate TCN on a simpler task: stacking an already-grasped yellow octagon (simple-stacking in Table 1(b)). The TCN performs reasonably well on this task, demonstrating its ability to imitate smooth trajectories, but having difficulty to imitate fine-grained grasping actions.

Task - Pouring. In this task, the robot needs to simultaneously translate and rotate the can to reach the desired orientation above the mug, as shown in 2nd row in Figure 3b. Using the same pouring demonstration, we additionally evaluate generalizability of VEGs by using a novel object with a different shape and texture during imitation (Figure 3b 3rd row). Trajectory optimization converges after 10 iterations, and the robot solves the task for all objects. The TCN baseline is able to move the can along the trajectory, but fails to rotate it to the right configuration for successful pouring.

Discussion/Limitations. Imitating the human hand critically improves performance during reaching and grasping motions in human demonstrations, which cannot be easily inferred from any object motion. While our method can deal with partial occlusions during imitation, we currently do not attempt tracking of fully occluded objects over a prolonged period of time. In light of this, a clear avenue for future work is learning to track through occlusions, e.g., by using temporal visual memory from which the reward graph can be extracted, as opposed to relying on visual frames only. Active vision can also be used both to undo occlusions during imitation, and to observe the demonstrator from the most convenient viewpoint for the imitator. Another limitation is the lack of any prior model information, which would accelerate the policy search. An interesting avenue for future work would be learning a model over the motion of such visual entities, and use it as a prior for better exploration during policy search.

5 Conclusion

We proposed encoding video frames in terms of visual entities and their spatial relationships, and we used this encoding to compute a perceptual cost function for visual imitation. Between end-to-end learning and engineered representations, we combine the best of both by incorporating important inductive biases regarding object fixation and motion saliency in our robot imitator. Experimental results on a real robotic platform demonstrate the generality and flexibility of our approach. Quoting the authors of [45], "just as biology uses nature and nurture cooperatively, we reject the false choice between "hand-engineering" and "end-to-end" learning, and instead advocate for an approach which benefits from their complementary strengths."

References

[1] P. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *Conference on Robot Learning*, 2018.

[2] P. Sermanet, C. Lynch, J. Hsu, and S. Levine. Time-Contrastive Networks: Self-Supervised Learning from Multi-view Observation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July:486–487, 2017. ISSN 21607516. doi:10.1109/CVPRW.2017.69.

[3] K. Y. Learning from examples: Imitation learning and emerging cognition. In *Humanoid Robotics and Neuroscience: Science, Engineering and Society.*, 2015.

[4] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[5] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *CoRR*, abs/1703.01703, 2017. URL http://arxiv.org/abs/1703.01703.

[6] S. Schaal. Is imitation learning the route to humanoid robots? 3(6):233–242, 1999. URL http://www-clmc.usc.edu/publications/S/schaal-TICS1999.pdf;http://www-clmc.usc.edu/publications/S/schaal-TICS1999-rep.pdf.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[9] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning. 2017. ISSN 00308870. doi:10.1016/j.aqpro.2013.07.003. URL http://arxiv.org/abs/1703.03078.

[10] T. Schmidt, R. Newcombe, and D. Fox. DART: Dense Articulated Real-Time Tracking. (1), 2015. ISSN 0094243X. doi:10.15607/rss.2014.x.030. URL http://www.roboticsproceedings.org/rss10/p30.pdf.

[11] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *CoRR*, abs/1703.02018, 2017. URL http://arxiv.org/abs/1703.02018.

[12] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2):21:1–21:35, Apr. 2017. ISSN 0360-0300. doi:10.1145/3054912. URL http://doi.acm.org/10.1145/3054912.

[13] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009. ISSN 0921-8890. doi:10.1016/j.robot.2008.10.024. URL http://dx.doi.org/10.1016/j.robot.2008.10.024.

[14] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *CoRR*, abs/1710.04615, 2017. URL http://arxiv.org/abs/1710.04615.

[15] C. L. Nehaniv and K. Dautenhahn. Imitation in animals and artifacts. chapter The Correspondence Problem, pages 41–61. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7. URL http://dl.acm.org/citation.cfm?id=762896.762899.

[16] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *CoRR*, abs/1709.10087, 2017. URL http://arxiv.org/abs/1709.10087.

[17] O. Kroemer and G. S. Sukhatme. Learning relevant features for manipulation skills using meta-level priors. *CoRR*, abs/1605.04439, 2016. URL http://arxiv.org/abs/1605.04439.

[18] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research*, 32(3):263–279, 2013.

[19] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. URL http://arxiv.org/abs/1603.08155.

[20] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1577–1584. IEEE, 2018.

[21] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016. URL http://arxiv.org/abs/1606.07419.

[22] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR*, abs/1509.06113, 2015. URL http://arxiv.org/abs/1509.06113.

[23] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *CoRR*, abs/1506.07365, 2015. URL http://arxiv.org/abs/1506.07365.

[24] T. Lesort, N. D. Rodríguez, J. Goudou, and D. Filliat. State representation learning for control: An overview. *CoRR*, abs/1802.04181, 2018. URL http://arxiv.org/abs/1802.04181.

[25] O. Rybkin, K. Pertsch, A. Jaegle, K. G. Derpanis, and K. Daniilidis. Unsupervised learning of sensorimotor affordances by stochastic future prediction. *CoRR*, abs/1806.09655, 2018. URL http://arxiv.org/abs/1806.09655.

[26] M. Sieb and K. Fragkiadaki. Data Dreaming for Object Detection: Learning Object-Centric State Representations for Visual Imitation. *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9, 2019. ISSN 21640580. doi:10.1109/humanoids.2018.8625007.

[27] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37(6), Nov. 2018.

[28] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, D. S. Phoenix, and D. George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *CoRR*, abs/1706.04317, 2017.

[29] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. *CoRR*, abs/1511.07404, 2015. URL http://arxiv.org/abs/1511.07404.

[30] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016. URL http://arxiv.org/abs/1612.00222.

[31] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. *CoRR*, abs/1511.05298, 2015. URL http://arxiv.org/abs/1511.05298.

[32] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, June 2016. doi:10.1109/CVPR.2016.110.

[33] C. Diuk, A. Cohen, and M. L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 240–247, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi:10.1145/1390156.1390187. URL http://doi.acm.org/10.1145/1390156.1390187.

[34] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.

[35] C. Devin, P. Abbeel, T. Darrell, and S. Levine. Deep Object-Centric Representations for Generalizable Robot Learning. 2017. URL http://arxiv.org/abs/1708.04225.

[36] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[37] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang. Vision-based Teleoperation of Shadow Dexterous Hand using End-to-End Deep Neural Network. URL https://arxiv.org/pdf/1809.06268.pdf.

[38] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[39] T. Schmidt, R. Newcombe, and D. Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017.

[40] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016.

[41] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[42] K. H. Lee, J. Lee, A. L. Thomaz, and A. F. Bobick. Effective robot task learning by focusing on task-relevant objects. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 2551–2556, 2009. doi:10.1109/IROS.2009.5353979.

[43] E. Hoffer and N. Ailon. Deep metric learning using triplet network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9370 (2010):84–92, 2015. ISSN 16113349. doi:10.1007/978-3-319-24261-3_7.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. 2015. ISSN 08866236. doi:10.1109/CVPR.2016.308. URL http://arxiv.org/abs/1512.00567.

[45] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, aglar Gülehre, F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

A Hyperparameters for Experimental Section

This is a comprehensive list of the hyperparameters used for our experiments. We note that this list might be non-exhaustive in the sense that we do not list hyperparameters which were copied over from their respective source implementation or if they are not detrimental to the performance our method to our knowledge. For example, we refrain from listing parameters such as the number of Anchor Scales used for training the Mask R-CNN architecture since this can be inferred from their source implementation.

Pushing **Task:**

1. Action space: 4-dimensional (XYZ + rotation)
2. Time step length: 0.4 sec
3. Episode length: 30
4. Cost weight hand edges: 1.0 (50.0 for direction-change variant)
5. Cost weight object edges: 1.0
6. Cost weight point edges: 0.0 (not used)
7. Number of rollouts per update cycle: 8
8. Controller gains: XYZ: 0.001, rotation: 0.02

Stacking **Task:**

1. Action space: 4-dimensional (XYZ + gripper state)
2. Time step length: 0.4 sec
3. Episode length: 30
4. Cost weight hand edges: 1.0
5. Cost weight object edges: 1.0
6. Cost weight point edges: 0.0 (not used)
7. Number of rollouts per update cycle: 8
8. Controller gains: XYZ: 0.001

Pouring **Task:**

1. Action space: 4-dimensional (XYZ + rotation)
2. Time step length: 0.4 sec
3. Episode length: 20
4. Cost weight hand edges: 1.0
5. Cost weight object edges: 1.0
6. Cost weight point edges: 1.0
7. Number of rollouts per update cycle: 10
8. Controller gains: XYZ: 0.00125, rotation: 0.02