Data Dreaming for Object Detection: Learning Object-Centric State Representations for Visual Imitation

Maximilian Sieb,[1] Katerina Fragkiadaki[1]

***Abstract*— We present a visual imitation learning method that enables robots to imitate demonstrated skills by learning a perceptual reward function based on object-centric feature representations. Our method uses the background configuration of the scene to compute object masks for the objects present. The robotic agent then trains a detector for the relevant objects in the scene via a process we call *data dreaming*, generating a synthetic dataset of images of various object occlusion configurations using only a small amount of background-subtracted ground truth images. We use the output of the object detector to learn an object-centric visual feature representation. We show that the resulting factorized feature representation comprised of per-object appearance features and cross-object relative locations enables efficient real world reinforcement learning that can teach a robot a policy based on a single demonstration after few minutes of training.**

I. Introduction

Imitation learning allows robots to acquire manipulation skills from human demonstrations. Using this approach, a robot can be trained quickly in an intuitive manner by non-roboticists to perform different tasks. The demonstrations can be provided to the robot using a number of techniques, including teleoperation and kinesthetic teaching. The most intuitive method for a human to provide a demonstration is through *visual demonstrations*, i.e., the human performs the task as they would naturally, and the robot observes the human using a camera. Although this method is easy for the human, it does not allow the robot to observe the actions and forces employed by the human to perform the task. The embodiment of the human and robot is different, which means that the human's motions and dynamics for performing the skill cannot be directly mapped to the robot's body.

The goal of manipulation tasks is to change the states of objects in the environment in a predictable manner. The robot therefore does not need to mimic the movements of the human exactly. Instead, it should imitate the state trajectories of the objects in the scene. We formulate this imitation problem with a reward function, wherein the robot receives more reward if its actions result in object states that are more *similar* to those observed during the human demonstrations. Using this reward function, the robot can employ reinforcement learning to learn policies for imitating the demonstrated manipulations in different situations. This approach allows the robot to learn suitable manipulation policies despite the differing embodiment of the human and robot.

[1]both authors are members of Carnegie Mellon University (CMU) {msieb, kfragki2}@andrew.cmu.edu

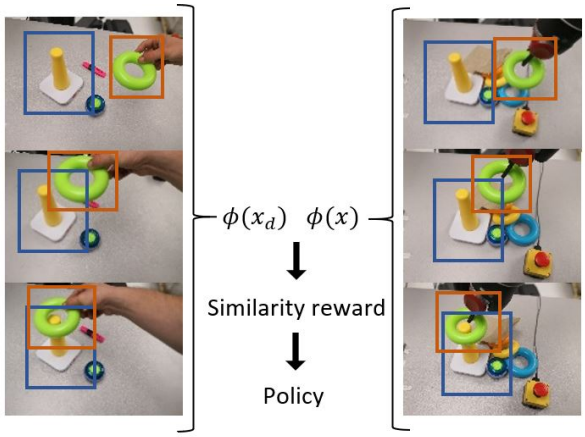

Fig. 1: **Learning similarities between human demonstration and robot trials.** The goal is to find a feature mapping ϕ that minimizes the distance of similar visual inputs in the embedding space.

The key problem with employing this approach is defining what it means for two manipulations to be more or less *similar*. If we directly compared the full pixel observations of the scenes, then even two distinct manipulations might be considered as similar because they were both performed in front of the same background. Instead, the robot should focus on the subset of relevant objects in the scene and ignore the irrelevant ones [19]. The similarity measure should also compare how the states of the relevant objects change over the course of the manipulation. These changes in state may correspond to local changes in the individual objects' appearances, and how they are arranged in the scene.

We propose learning state representations for visual imitation with a structural bias regarding object attention. Given an input video frame x, the proposed feature representation $\phi(x)$ is comprised of appearance features of each of the relevant objects in the scene, as well as the relative object distances in 3D. The proposed object-centric state representation is made possible via on-the-fly training of state-of-the-art neural object detectors [13] for novel objects, not present in large scale human annotated datasets, such as MS-COCO [21]. The detector is trained instead using synthetic data "dreamed" from unoccluded RGB masks of the relevant objects which are obtained through background subtraction. The masked images are then pasted onto a base image, visually augmenting the pasted images. While not generating completely realistic images, the synthetically generated images suffice to learn accurate detectors for the

relevant objects.

During the imitation process, the learner detects each relevant object in the scene and extracts object appearance features from each detected object bounding box as well as the relative distances of the 3D-centroids of the objects in the scenes. The object appearance features are learned using unsupervised time-contrastive losses [31], in order to capture subtle changes in the objects' appearances over time. The proposed object-factorized state representation is used in a trajectory optimization method [7], both to represent the state, as well as the cost, as Euclidean distance between the state of the demonstrator and the state of the imitator.

This paper attempts to close the gap between deep object detectors used in computer vision and machine perception based approaches in robotic learning. Due to the fact that object instances encountered by a robotic agent are often task-specific and not part of the categories labelled in commonly available computer vision datasets, the successful application of deep object detectors in this settings is often challenging. Instead, frame-wide representations are preferred, both for reinforcement learning [22], [3] and for visual imitation [31]. We show that generating a synthetic dataset based on a few ground truth examples, commonly referred to as data dreaming, is sufficient to train on-the-fly object detectors. Even though they do not detect each target object in *all* possible contexts, they are powerful enough in the current scene to track the object through partial occlusions.

The proposed framework was evaluated both in a simulator on several pick-and-place tasks, as well as on a Baxter robot putting rings onto a pole and rotating an object. Our experiments suggest object-centric state representations outperform frame-wide ones [31], which do not introduce any structural biases regarding object-centric attention. Agreeing with authors of [5], incorporating useful structural biases can greatly accelerate learning of neural architectures and alleviate the need of obtaining a large enough dataset to train these architectures. Furthermore, structural bias via placing hard attention on relevant objects greatly aids the interpretability of the algorithm compared to frame-centric end-to-end approaches.

II. RELATED WORK

a) Visual imitation learning: Imitation learning concerns the problem of acquiring skills by observing demonstrations. It has been considered an integral part in the field of robotics, with the potential to highly expedite trial-and-error learning [30]. However, the vast majority of previous approaches assume that such demonstrations are given in the workspace of the agent, (e.g., through kinesthetic teaching or teleoperation in the case of robots [16], [4]) and the actions/decisions of the demonstrator can be imitated directly, either with behavioral cloning (BC) [36], inverse reinforcement learning (IRL) [25], or generative adversarial imitation learning (GAIL) [14].

The problem of mimicking humans based on visual information is challenging due to the difficulet visual inference needed for fine-grained activity understanding [32]. In this case, imitation requires a mapping between observations in the demonstrator space to observations in the imitator space [24]. Numerous works circumvent the difficult perception problem using special instrumentation of the environment to read off object and hand poses during video demonstrations, such as AR tags, and use rewards based on known 3D object goal configurations [28]. Other works use hand-designed reward detectors that only work on restrictive scenarios [23].

Recent approaches bypass explicit state estimation and attempt to imitate the demonstrator's behaviour by directly matching latent perceptual states to which the input images are mapped. Numerous loss functions have been proposed to learn such an image or image-sequence encoding. One single RGB frame or an entire sequence of frames is encoded into an embedding vector using a combination of forward and inverse dynamics model learning in [27], [1], multiview invariant and time-contrastive metric learning in [31], or reconstruction and temporal prediction objectives in [9], [35]. [20] provides an overview of common loss metrics and inductive biases for state representation learning.

Having obtained this latent state space, imitation is carried out by iterative application of the inverse model in [27], [1], or via trajectory optimization in [31], where the state is the combination of the latent visual state embedding vector and the robot configuration. Our work is most similar to [31], in that we use a similar trajectory optimization method [7] for imitating a human visual demonstration and similar unsupervised loss functions for visual feature learning. However, we apply such metric learning to features collected within each relevant object bounding box, as opposed to a video frame, i.e., we share neural feature extraction weights across objects in the scene.

Utilizing graph encodings of a visual scene in terms of objects or parts and their pairwise relations has been found beneficial for generalization of action-conditioned scene dynamics [11], [6], and body motion and person trajectory forecasting [17], [2]. In contrast to the work of [9] which learns to represent an RGB image in terms of spatial coordinates of few keypoints, we employ explicit attention only to relevant objects, and preserve their correspondence in time, i.e., the detectors *bind* with specific objects in the scene.

b) Synthetic data augmentation: Synthetic data augmentation has been very successful in multiple machine perception tasks [26], [12], [34], [29]. In [12], images with cross-person occlusions are generated to help train occlusion-aware detectors. Context-specific data augmentation was used in [18] to train neural networks for object video segmentation. The work of [34] showed that domain randomization, namely augmenting textures of the background and the objects, even in a non-visually plausible way, helps in training successful detectors that work on real images.

III. OBJECT-CENTRIC PERCEPTUAL REWARD LEARNING

A. Problem Statement

Our goal is to learn a policy that imitates a human based on a single demonstration of a manipulation task. We formulate

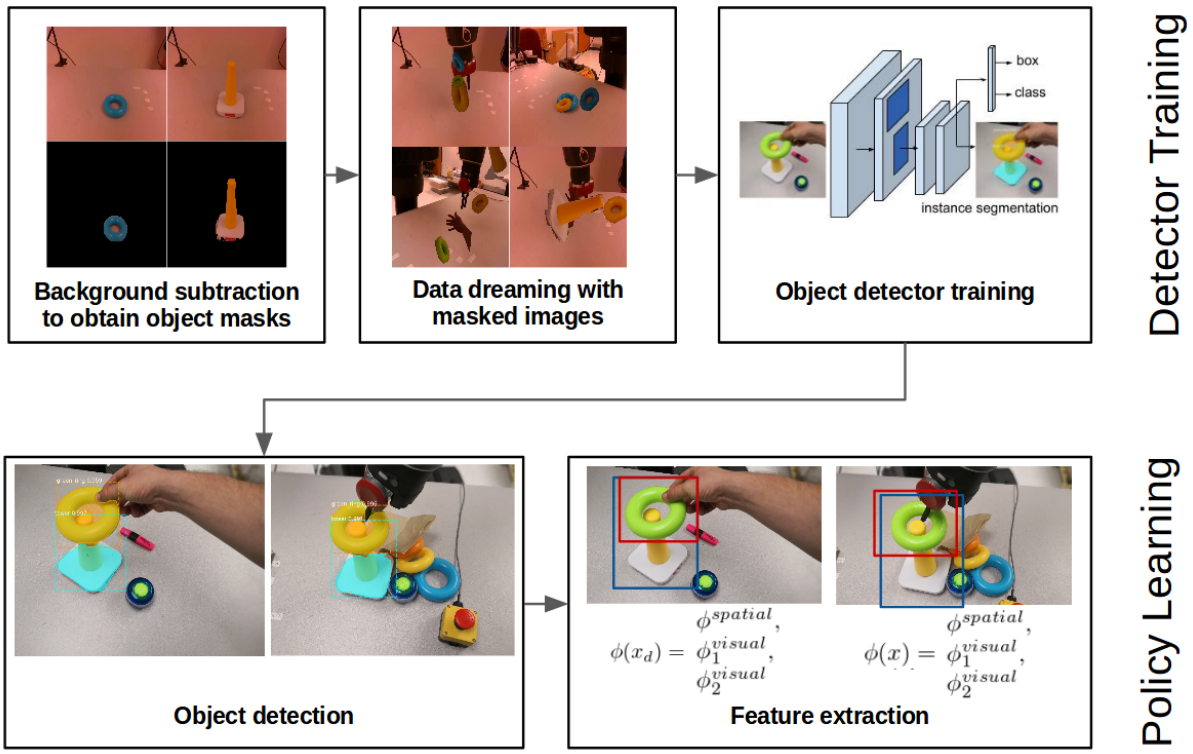

Fig. 2: **Overview.**Using a few frames in which we can obtain ground truth masks through standard background subtraction and connected components, we create synthetic data and train an object detector that is robust to occlusions. We then collect human demonstrations and detect bounding boxes of all relevant objects in the scene, fine-tuning our detector network in case of missed detection by generating more synthetic training data. During inference, we detect the bounding boxes of all relevant objects in the scene, compute appearance features using our trained feature network and object relative distance information directly from the bounding box centroids and use this feature representation for learning policies on the robot. The combined object appearance and relative distance features are used to guide imitation learning on a real robot.

this as a reinforcement learning problem, wherein the robot is given a reward based on the similarity between the visual observations $\mathbf{x}$ resulting from its execution of the task, and the visual observations $\mathbf{x}_d$ obtained during the human's visual demonstration of the task [8]. We then define the step-wise reward function similar to [31] as:

$$R(\phi(\mathbf{x}), \phi(\mathbf{x}_d)) = -\alpha \frac{1}{2} ||\phi(\mathbf{x}) - \phi(\mathbf{x})_d||_2^2 - \beta \sqrt{||\phi(\mathbf{x}) - \phi(\mathbf{x})_d||_2^2 + \gamma}, \quad (1)$$

where ϕ is a function that maps the observation $\mathbf{x}$ into a latent embedding space in which high perceptual similarity between visual frames is demonstrated as small Euclidean distance between their corresponding feature embeddings and α, β and γ are hyperparameters that we set to $\alpha = 10$, $\beta = 0.001$ and $\gamma = 10^{-5}$ for all experiments. This loss is in essence a smooth version of the L1-Loss, and is less sensitive towards outliers than the L2-Loss.

The observation encoding function ϕ is not provided to the robot and must be learned. Inverse reinforcement learning methods attempt to learn such observation encoding function from a large number of expert demonstrations. In this work, we opt for an unsupervised learning method instead. The observation encoding function should focus on the relevant objects in the scene and ignore irrelevant ones. We learn detectors of the relevant objects and use them to provide instance specific attention bounding boxes in both demonstration and imitation videos.

In the following sections, we describe object detector training and learning of object-centered appearance features. Given a learned observation encoding function $\phi(\mathbf{x})$, we will use trajectory optimization [7] to compute a time-dependent parameterized policy $\pi_t(u|\phi(\mathbf{x}), \mathbf{x_r}; \theta)$ that defines the distribution over actions u given the current encoded observation $\phi(\mathbf{x})$ and robot configuration $\mathbf{x_r}$ and maximizes the expected return of the entire trajectory $J(\theta) = \mathbb{E}_{u \sim \pi_t(u|\mathbf{x}, \mathbf{x_r}; \theta)}[\sum_t R(\phi(\mathbf{x}), \phi(\mathbf{x}_d))]$.

B. Object Detection via Data Dreaming

To extract object-centric features, we need to first predict 2D bounding boxes that capture each of the relevant objects in the scene in the visual demonstrations and during robot execution. We do not have bounding box annotations for the objects present in the scene. It is not possible to use a pretrained detector when the objects the robot encounters are not contained in the datasets on which the detector was trained, e.g., in Fig. 2, our robot task involves toy rings and a yellow pole, categories not present in MS-COCO. Even if the

object category is part of a category in a human-annotated dataset, we still need to train *instance-specific* detectors in order to reliably track the objects during manipulation (e.g., different instances of the same object class cannot be uniquely identified by the detector if it has not been trained on these specific object instances before, greatly exacerbating the problem of accurately tracking them.)

Manually generating the masks and bounding boxes for the relevant objects in a variety of configurations in order to fine-tune the detector is a tedious process. Our main insight in this work is to create a sufficiently large synthetic dataset on-the-fly by translating, scaling, rotating and changing the pixel intensity of RGB masks of the objects in the scene, as shown in Fig. 2. Objects' RGB masks are automatically obtained using background subtraction in the initial frame, where we assume the objects are not occluding one another. Each image is also given a label, e.g., "blue ring", to identify the relevant object in the scene. In the synthetically generated images, the segmented object masks will often overlap with one another. Such overlaps help the detector to be robust under partial occlusions, still predicting the correct *amodal* bounding box, i.e. predicting the entire bounding box even if the corresponding object is only partially visible. In our case, we fine-tune Mask R-CNN [13] to predict the synthetically generated object labels, namely masks and bounding boxes.

We alternate between training our object detector creating synthetic image augmentations. Specifically, whenever the robot fails to detect an object, we use the last frame in which it confidently detected the object, using the classification score as the confidence metric. The robot then uses the mask from this high-confidence frame to create a new cropped object image. This cropped object frame is then used to create additional synthetic images using the same transformations as before to fine-tune the detector network. In this manner, the robot continues to expand its training set over time to fill in gaps in the training distribution. While complete object occlusion is possible, this can mostly be avoided by choosing an amenable camera angle or using multiple cameras and switching the viewpoint in case of total object occlusion during a trial.

Given multiple objects present in the scene, we need a way to indicate to our agent which ones are relevant for the task. In our case, we provided labels to the objects within the training set, and thus, we can specify the label of the target objects in the scene. The output of our visual detector is thus an ordered set of bounding boxes and segmentation masks that correspond to the set of relevant objects in the scene.

C. Object-Centric Appearance Feature Learning

Given the predicted object bounding boxes, our feature representation is comprised of two parts, what and where: per object appearance features ϕ^{visual} and cross-object 3D spatial arrangements $\phi^{spatial}$. An Intel RealSense camera provides the robot with Cartesian X, Y, and Z coordinates for each pixel in the RGB-D image. Using the object segmentation mask predicted by our detector, the robot extracts the point cloud corresponding to each object. The 3D position of the object is then given by the centroid of the X, Y, Z point cloud. We assume that we are supplied a task-specific pre-defined ordering of the relevant objects.

Depending on the task to be solved, different inductive biases can be more or less appropriate. As for the object appearance features in manipulation-based tasks, these should ideally be highly discriminative of the objects' temporal changes relevant to the task success. We choose to train object appearance features using time-contrastive metric learning proposed in [31]: The authors learn a feature mapping that pulls temporally close RGB frames together in the embedding space, and pushes temporally far frames away from each other. In this work, we apply time-contrastive metric learning using object tubes (the temporal sequence of the object-specific bounding boxes), as opposed to the full-frame video sequence. Specifically, we use a triplet loss as introduced in [15], ensuring that temporally close neighbours in each object tube are closer to each other in the embedding space than any temporally distant pair of boxes within the same object tube. Formally, given any object-specific object tube, this condition reads:

$$||\psi(\mathbf{x_b}^a) - \psi(\mathbf{x_b}^+)||_2^2 + \alpha < ||\psi(\mathbf{x_b}^a) - \psi(\mathbf{x_b}^-)||_2^2, \quad (2)$$

where $\mathbf{x_b}$ denotes the RGB sequence of bounding boxes for a specific object (an example is shown in Fig. 2 under step 2 of the pipeline), α is the margin which we set to 2.0 during our experiments, $\mathbf{x}^a$ is a randomly sampled anchor frame, $\mathbf{x}^+$ is a frame that is temporally close to the anchor (positive sample), and $\mathbf{x}^-$ is a frame that is temporally distant from the anchor (negative sample). The positive sample is only drawn within a certain range around the anchor frame (in our case within 4 frames), and the negative sample is only drawn outside of a certain range (in our case at least 8 frames apart). If the scene is static, for example, then negative frames are just as similar to the anchor frame as positive frames, greatly hindering training. In our experiments, however, all relevant objects underwent at least partial visual change during the demonstration avoiding the aforementioned scenario.

Training our object-centric time-contrastive network allows the robot to learn features that are useful for detecting visual changes within an object's bounding box that spatial relationships cannot differentiate, including interaction with hands and robot grippers. We train the network on multiple views of the human demonstration. The time-contrastive network is implemented using the same architecture as in [31], and we also use pre-trained weights of the Inception-v3 model [33]. Furthermore, we extend the network architecture by also incorporating depth information as an input. To do so, we feed the depth as a separate input channel into the network and run it through three convolutional layers followed by a max pool layer, a spatial softmax layer and finally a fully connected layer to output a 16-dimensional vector. We concatenate this output and the output of the RGB stream (which outputs a 16-dimensional feature vector in our case) to obtain a 32-dimensional feature vector as the overall output of the network, which is computed for

all relevant objects. We then obtain the final observation encoding representation by concatenating the 32-dimensional feature vectors for each object ϕ_i^{visual} and the cross-object 3D distances $\phi_i^{spatial}$ between all objects relative to the anchor object.

D. Policy Learning from Demonstration

Given the reward function in Eq. 1, we learn a policy for imitating the demonstrated manipulation using PILQR [7], a state-of-the-art trajectory optimization method that combines, in a principled manner, the model-based generalization of linear quadratic regulators together with path integral policy search's flexibility and ability to handle non-linear dynamics. We learn a separate time-dependent policy $\pi_t(u|x;\theta) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{\Sigma}_t)$ for each set of objects and initial object locations. This policy is a time-dependent linear feedback controller, as used in linear quadratic optimal control. The time-dependent control gains are learned by alternating model-based and model-free updates, where the dynamical model $p(x_t|, x_{t-1}, u_t)$ of the *a priori* unknown dynamics is learned during training time. For more detail, please refer to [7]. The actions 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 gripper is always pointing downwards in our experiments. The input to the policy consists of the joint angles, end-effector position, the object location and appearance features resulting in an $n_{joints} + N * 32 + (N-1) * 3$ dimensional state space for N objects involved where n_{joints} denotes the number of joints of the robot.

IV. EXPERIMENTS

Our experiments aim to answer the following questions:

1) How much does the proposed object-centric RGB encoding benefit reward shaping over frame-centric feature encodings of previous works?
2) How well do appearance and location features contribute to reward shaping?
3) How much does the proposed object-centric RGB encoding benefit policy learning for sample efficient visual imitation over previous alternatives?

A. Reward Shaping

For a comparison between frame-centric single-view [31] and the proposed object-centric image encoding, we compare in Fig. 3 the reward value of Eq. 1 as produced by the two feature functions, while using only the object appearance features and not cross-object distances in our proposed feature function. The task is defined by rotating a cylindrically shaped object around one of its principal axis. In Fig. 3 first row we show the human demonstration, in the second row we show a correct robot imitation and in the third row we show a wrong behaviour. In both cases, the feature encoding was trained using a video sequence in which the tower was moved by a human. Note the object-centric appearance features, here involving the one object of interest (yellow tower), are more discriminative of the correct task execution. The object-centric features are also more attentive towards minor changes in appearance, such as how the robot is holding the object. For purely translational tasks, such as stacking or moving objects, the object location features are often sufficient and adding appearance features is not imperative for the success of the learner. As an example task, we perform an experiment of putting a ring on top of a cylindrical tower object as demonstrated by a human. We show in Fig. 5 the reward curves for only the spatial arrangement features that generalize well across objects (blue to green ring) and across human and robot demonstrator. Establishing cross-object correspondences thus permits strong generalization for free using such appearance agnostic features, that do not require immense amounts of data to be trained in order to forget the appearance of the original objects.

B. Policy Learning

We evaluate the suitability of our method for learning robot policies both in simulation and in real world on the following tasks:

1) **Ring-on-Tower**: The robot needs to learn how to put a ring on top of a cylindrical tower given a human demonstration, as seen in Fig. 5.
2) **Block-on-Block**: The robot needs to learn how to stack a cube onto another cube given a video of a VR demonstration of the robot agent.
3) **Block-in-Bowl**: The robot needs to learn how to place a cube inside a bowl given a video of a VR demonstration of the robot agent.

We evaluate the following architectures:

1) *ours*: detector-based object-centric feature learning
2) *TCN*: frame-centric TCN using RGBD input trained on demonstration data
3) *blob detector*: off-the-shelf blob detector generating encodings the same way as for our method

For the first experiment, we use a Rethink Robotics Baxter while for the other two tasks we work with the Pybullet simulation engine using the provided KUKA manipulator model. For each task, we provide the policy learner with a single successful demonstration of the task and a detector network trained on the relevant objects of the scene.

For all experiments, we employ the robot's task space controller allowing end-effector movements in the x, y, z-direction as well as rotation around the vertical axis. Furthermore, we assume the availability of a grasping primitive that succesfully grasps the relevant object. The control frequency is 2 Hz. The inference time of the entire pipeline is roughly 0.4 seconds, including the object detector and feature extractor. Since we used a *ResNet101* backbone in our Mask R-CNN architecture, we expect much faster inference time for less complex backbone models. For the experiments, we use a fixed episode length, at which point the robot's episode is automatically terminated. For the policy learning, we use an existing implementation of

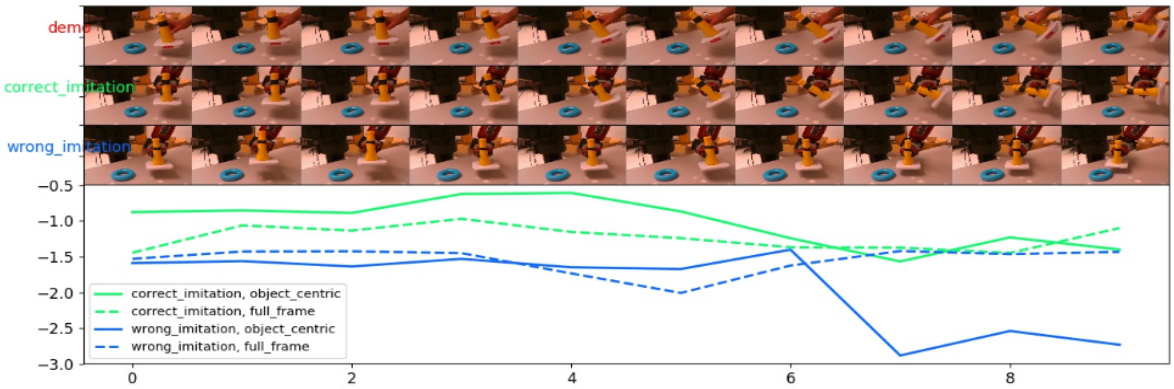

Fig. 3: **Reward comparison of object-centric and frame-centric time-contrasted visual features.** In the proposed object-centric feature representation, while the robot's correct execution is attributed with a high reward, wrong execution is penalized heavily. Frame-centric features cannot discriminate between correct and wrong execution. Note that the robot's hand has not been seen at all during training time.

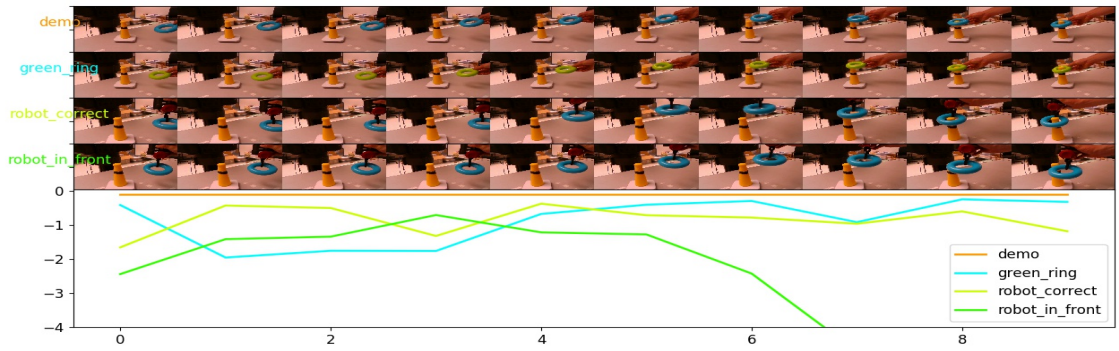

Fig. 4: **Reward comparison of spatial arrangement features.** The reward decreases once the ring is not stacked on top of the tower. The reward also generalizes well to a ring of different color and smaller size.

PILQR [10] as described in section III.D. To stay near the dynamical models that were learned as part of PILQR, we define a maximum movement of 3 cm in each principle direction for each time step and a 10 degree rotational movement limit.

In addition to the time needed to learn the policy through PILQR, we trained the object detector on the synthetic dataset for approximately 20 minutes beforehand. This detector pre-training only has to be done if new object classes are introduced and can be reused across different tasks.

1) Ring-on-Tower Task in Real World: We set the episode length to 10 for this task. For the PILQR learning, the robot attempted the task 7 times between each policy update. During each time step, the detector network generates bounding boxes and masks of the relevant objects in the scene. The robot solves the ring-stacking task after only 3 iterations of PILQR (taking around 5 minutes) after having been shown a single human demonstration, as shown in Fig. 6.

We trained the time-contrastive network on three sequences from three different views each, namely the human demonstration itself as well as two video sequences performing arbitrary movements with the ring and tower. Although we use multiple views, the network is trained using each view separately and during test time only one view is used from the robot's perspective.

For every iteration in PILQR, the robot has to reset to the same initial condition. This initial condition, however, does not need to coincide with the demonstrations's initial condition and can undergo arbitrary translational variations due to our object-centric location-invariant feature representation.

The task was neither solved using only a full-frame time-contrastive feature representation nor using a *blob detector* given the same training and test setup.

2) Block-on-Block Task in Simulation: We set the episode length to 20 for this task. For the PILQR learning, the robot attempted the task 15 times between each policy update. During each time step, the detector network generates bounding boxes and masks of the relevant objects in the scene. The robot solves the task after only one iteration

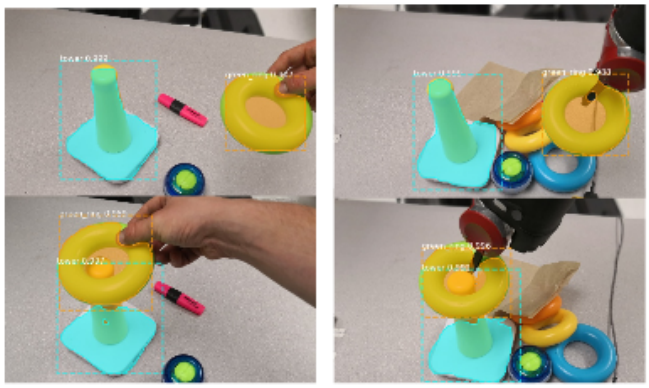

Fig. 5: **Real robot task execution**. *Left*: Human demonstration of ring-on-tower task. *Right*: Robot executing the demonstrated task. The bounding boxes extend even across the occluded parts of the object.

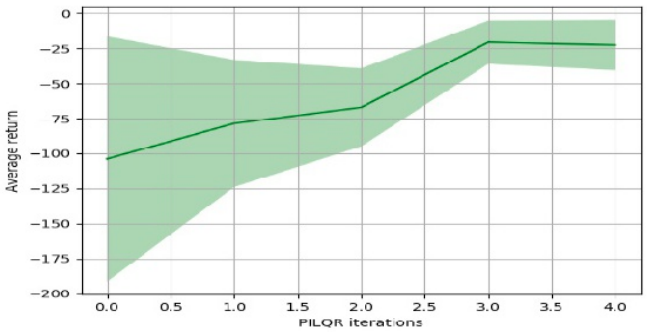

Fig. 6: **Learning progress of real world ring-on-tower task.** We see that the robot quickly learns to imitate the demonstrated behaviour. Note that the reward does not converge to zero since the robot does not start in the exact same initial position as the learner and is therefore not able to perfectly match the entire object trajectory for all time steps.

of PILQR, having been shown a single demonstration, as shown in Fig. 7. There, we see the average success rate over multiple tries with varied starting positions of both objects, randomly sampled within a 10 cm radius of the original demonstration position. We experienced that the spatial feature encodings are sufficient enough to solve this task and did not see any deterioration in performance when only using those. The *blob* is also able to solve the task, it does so with much more variance in the execution. This is due to the less accurate estimate of the correct depth value because our method provides exact object masks with accurate object depth estimates. For the *blob detector*, heuristic methods such as color-based segmentation have to be used to obtain object masks which fluctuated more during our experiments. The *TCN* baseline, which we trained on 100 views of that single demonstration, fails to solve the task.

3) Block-in-Bowl Task in Simulation: We set the episode length to 20 for this task. With respect to the policy learning,

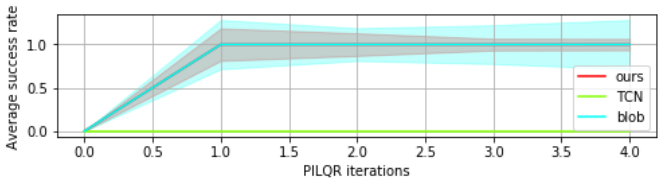

Fig. 7: **Average success rate of block-on-block task.** The robot learns to stack the block for both our method and the *blob detector* baseline. It fails to do so for the *TCN* baseline. Our method imitates the original trajectory more closely due to to higher precision of the trained object detector.

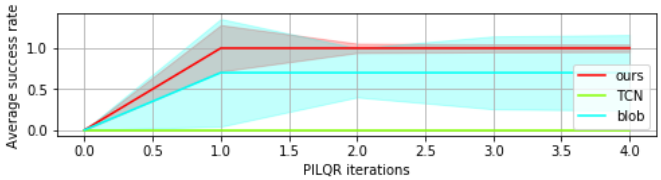

Fig. 8: **Average success rate of block-in-bowl task with occluders.** The robot learns to place the block into the bowl for both our method and the *blob detector* baseline. It fails to do so for the *TCN* baseline. Due to the occlusions introduced into the setup, the *blob detector* shows much more variance because the detections are further off with respect to the ground truth state compared to our trained object detector.

the robot attempted the task 15 times between each policy update. During each time step, the detector network generates bounding boxes and masks of the relevant objects in the scene. The robot solves the task after only one iteration of PILQR, having been shown a single demonstration, as shown in Fig. 8. There, we see the average success rate over multiple tries with varied starting positions of both objects, randomly sampled within a 10 cm radius of the original demonstration position.

Because we include partial occlusions in the form of distractor objects and artificial blacking out of pixels, the performance of the *blob detector* deteriorates, while the trained detector network still manages to solve the task robustly. The task was not solved using the *TCN* baseline, which was trained on 100 views of the provided demonstration.

V. Conclusion

We presented an object-centric feature learning method for state representation and reward learning for visual imitation. The learned state representation encodes how the relevant objects are arranged in the scene, and how the state of the objects changes during the manipulation. The features are learned in an efficient manner by training state-of-the-art object detector architectures using dreamed data by augmenting object RGB-masks obtained automatically through background subtraction. We showed the effectiveness of the learned state representation by employing it on a trajectory optimization method and showed a Baxter robot could learn a task within a few minutes using a single demonstration.

Our approach currently has the following limitations: first, it cannot handle full object occlusions, and no advanced tracking is employed in any way. We rely on per-frame object detections provided by our iterative data dreaming and tracking-by-detection. Secondly, the within-object appearance features are far less discriminative of task completion than the cross-object spatial features. In future work, we will focus on end-to-end visual object tracking in manipulation environments, and we plan to explore multiscale entity graphs, where nodes not only represent single objects as in thise work, but also sub-parts of individual objects.

VI. Acknowledgements

We would like to sincerely thank Oliver Kroemer for fruitful discussions and insights during the course of this research, as well as his extensive input on preparation of this manuscript. We would also like to thank Deepika Bablani for sharing her experience regarding training object detectors on-the-fly and Yihe Tang for providing virtual reality demonstration for visual imitation in simulation. Maximilian Sieb was partially supported by the German Academic Exchange Service and the German Academic Scholarship Foundation during the course of this research.

References

[1] 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.

[2] 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.

[3] S. Amarjyoti. Deep reinforcement learning for robotic manipulation-the state of the art. *arXiv preprint arXiv:1701.08878*, 2017.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21(5):388–403, 2013.

[9] 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.

[10] C. Finn, M. Zhang, J. Fu, X. Tan, Z. McCarthy, E. Scharff, and S. Levine. Guided policy search code implementation, 2016. Software available from rll.berkeley.edu/gps.

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

[12] G. Ghiasi, Y. Yang, D. Ramanan, and C. Fowlkes. Parsing occluded people. In *CVPR*, 2014.

[13] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[14] J. Ho and S. Ermon. Generative adversarial imitation learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016.

[15] 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.

[16] 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.

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

[18] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. *CoRR*, abs/1703.09554, 2017.

[19] O. Kroemer and G. S. Sukhatme. Learning relevant features for manipulation skills using meta-level priors. *CoRR*, abs/1605.04439, 2016.

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

[21] 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.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[23] 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.

[24] C. L. Nehaniv and K. Dautenhahn. Imitation in animals and artifacts. chapter The Correspondence Problem, pages 41–61. MIT Press, Cambridge, MA, USA, 2002.

[25] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[26] D. Park and D. Ramanan. Articulated pose estimation with tiny synthetic videos. *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 58–66, 2015.

[27] 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.

[28] 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.

[29] F. Sadeghi and S. Levine. (cad)$ˆ2$rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016.

[30] S. Schaal. Is imitation learning the route to humanoid robots? 3(6):233–242, 1999.

[31] 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.

[32] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *CoRR*, abs/1703.01703, 2017.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. 2015.

[34] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017.

[35] 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.

[36] 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.