Robust Object Pose Tracking for Robotics

Chelsea Finn Justin Fu Nopphon Siranart

***Abstract*—Policy search methods based on reinforcement learning and optimal control can allow robots to automatically learn a wide range of tasks. However, practical applications of policy search tend to require the policy to be supported by hand-engineered components for perception, state estimation, and low-level control. We propose a method for learning policies that map raw, low-level observations, consisting of joint angles and camera images, directly to the torques at the robot's joints. The policies are represented as deep convolutional neural networks (CNNs) with 92,000 parameters. The high dimensionality of such policies poses a tremendous challenge for policy search. To address this challenge, we develop a sensorimotor guided policy search method that can handle high-dimensional policies and partially observed tasks. We use BADMM to decompose policy search into an optimal control phase and supervised learning phase, allowing CNN policies to be trained with standard supervised learning techniques. This method can learn a number of manipulation tasks that require close coordination between vision and control, including inserting a block into a shape sorting cube, screwing on a bottle cap, fitting the claw of a toy hammer under a nail with various grasps, and placing a coat hanger on a clothes rack.**

I. Introduction

Reinforcement learning and policy search methods hold the promise of allowing robots to acquire new behaviors through experience. They have been applied to a range of robotic tasks, including manipulation [**?** **?**] and locomotion [**?** **?** **?** **?**]. However, policies learned using such methods often rely on a number of hand-engineered components for perception and low-level control. The policy might specify a trajectory in task-space, relying on hand-designed PD controllers to execute the desired motion, and a policy for manipulating objects might rely on an existing vision system to localize these objects [**?**]. The vision system in particular can be complex and prone to errors, and its performance is typically not improved during policy training, nor adapted to the goal of the task.

We propose a method for learning policies that directly map raw observations, including joint angles and camera images, to motor torques. The policies are trained end-to-end using real-world experience, optimizing both the control and perception components on the same measure of task performance. This allows the policy to learn goal-driven perception, which avoids the mistakes that are most costly for task performance. Learning perception and control in a general and flexible way requires a large, expressive model. Our policies are represented with convolutional neural networks (CNNs), which have 92,000 parameters and 7 layers. Deep CNN models have been shown to achieve state of the art results on a number of supervised vision tasks [**?** **?** **?**], but sensorimotor deep learning remains a challenging prospect. The policies are extremely high dimensional, and the control task is partially observed, since part of the state must be inferred from images.

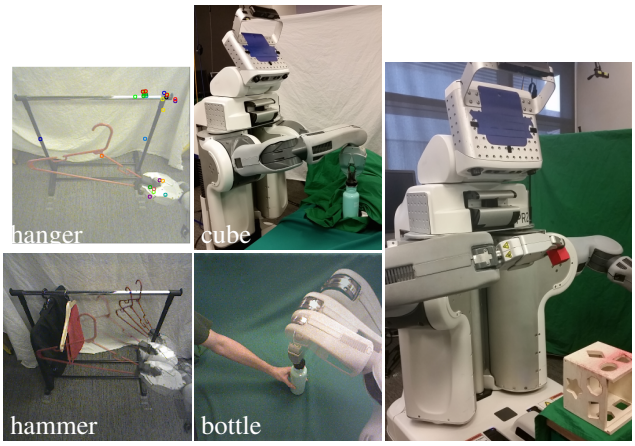


Fig. 1: Our method learns visuomotor policies that directly use camera image observations (left) to set motor torques on a PR2 robot (right).

To address these challenges, we extend the framework of guided policy search to sensorimotor deep learning. Guided policy search decomposes the policy learning problem into two phases: a trajectory optimization phase that determines how to solve the task in a few specific conditions, and a supervised learning phase that trains the policy from these successful executions with supervised learning [**?**]. Since the CNN policy is trained with supervised learning, we can use the tools developed in the deep learning community to make this phase simple and efficient. We handle the partial observability of visuomotor control by optimizing the trajectories with full state information, while providing only partial observations (consisting of images and robot configurations) to the policy. The trajectories are optimized under unknown dynamics, using real-world experience and minimal prior knowledge.

The main contribution of our work is a method for end-to-end training of deep visuomotor policies for robotic manipulation. We propose a partially observed guided policy search algorithm that can train high-dimensional policies for tasks where part of the state must be determined from camera images. We also introduce a novel CNN architecture designed for robotic control, shown in Figure 2. The vision layers of this CNN are designed for localizing points of interest in an image, unlike standard vision architectures that discard locational information to induce translational invariance [**?**]. We evaluate our method by learning policies for inserting a block into a shape sorting cube, screwing a cap onto a bottle, fitting the claw of a toy hammer under a nail with various grasps, and placing a coat hanger on a rack (see Figure 1). Our results demonstrate clear improvements in consistency and generalization from training visuomotor policies end-to-end,

when compared to using the poses or features produced by a CNN trained for 3D object localization.

II. Related Work

Reinforcement learning and policy search have been applied in robotics for playing games such as table tennis [**?**], object manipulation [**? ?**], and locomotion [**? ? ? ?**]. Several recent papers provide surveys of policy search in robotics [**? ?**]. Such methods are typically applied to one component of the robot control pipeline, which often sits on top of a hand-designed controller, such as a PD controller, and accepts processed input, for example from an existing vision pipeline [**?**]. Our method trains policies that map visual input and joint encoder signals directly to the torques at the robot's joints. By learning the entire mapping from perception to control, the perception layers can be adapted to optimize task performance.

We represent our policies with convolutional neural networks (CNNs). CNNs have recently achieved dramatic improvements on a number of vision benchmarks [**? ? ?**]. Most applications of CNNs focus on classification, where locational information is intentionally discarded by means of successive pooling layers [**?**]. Applications to localization typically either use a sliding window to localize the object, reducing the task to classification [**?**], perform regression to a heatmap of manually labeled keypoints [**?**], requiring precise knowledge of the object position in the image and camera calibration, or use 3D models to localize previously scanned objects [**? ?**]. We use a novel CNN architecture that automatically learns feature points without any supervision beyond the information from the robot's encoders and camera.

Unlike with visual recognition, applications of deep networks to robotic control have been comparatively limited. Backpropagation through the dynamics and the image formation process is impractical, since they are often non-differentiable, and such long-range backpropagation leads to extreme numerical instability. The high dimensionality of the network also makes reinforcement learning very difficult [**?**]. Pioneering early work on neural network control used small, simple networks [**? ?**], and has largely been supplanted by methods that use carefully designed policies that can be learned efficiently with reinforcement learning [**?**]. More recent work on sensorimotor deep learning has tackled simple task-space motion [**?**] and used unsupervised learning to obtain low-dimensional state spaces from images [**?**], but such methods are limited to tasks with a low-dimensional structure. CNNs have also been trained to play video games with temporal difference learning and Monte Carlo tree search [**? ?**]. However, such methods have only been demonstrated on discrete, synthetic domains, and require an impractical number of samples for real-world robotic applications. Our method is sample efficient, requiring only minutes of interaction time. To the best of our knowledge, this is the first method that can train deep visuomotor policies for complex, high-dimensional manipulation skills with direct torque control.

Learning visuomotor policies end-to-end introduces two key challenges: partial observability and the high dimensionality of the policy. We tackle these challenges using guided policy search. In guided policy search, the policy is optimized using supervised learning, which scales gracefully with the dimensionality of the function approximator. The training set for this supervised learning procedure can be constructed from example demonstrations [**?**], trajectory optimization under known dynamics [**? ? ?**], and trajectory-centric reinforcement learning methods that operate under unknown dynamics [**? ?**], which is the approach taken in this work. We propose a new, partially observed guided policy search method based on the Bregman alternating directions method of multipliers (BADMM) that makes it practical to train complex, generalizable policies under partial observation.

The goal of our approach is also similar to visual servoing, which performs feedback control on feature points in a camera image [**? ? ?**]. However, our visuomotor policies are entirely learned from real-world data, and do not require feature points or feedback controllers to be specified by hand. This gives our method considerable flexibility in choosing how to use the visual signal. Furthermore, our approach does not require any sort of camera calibration, in contrast to many visual servoing methods (though not all – see e.g. [**? ?**]).

III. Overview

The aim of our method is to learn a policy $(|)$ that specifies a distribution over actions conditioned on the observation , which includes a camera image and the configuration of the robot. The policy parameters are optimized to minimize a cost function $(,)$ over the course of a fixed-length episode. The actions are the motor torques, and the state includes the known robot configuration as well as (for example) the target position for an object placement task or the grasp pose. The latter information is not observed directly by the policy, and must be inferred from the camera image. We represent $(|)$ as a Gaussian, with the mean given by a nonlinear function approximator. Since this function approximator needs to operate directly on raw images, we use convolutional neural networks (CNNs), which have enjoyed considerable success in computer vision [**?**]. The architecture of our CNN is shown in Figure 2. This network has 7 layers and around 92,000 parameters, which presents a tremendous challenge for standard policy search methods [**?**].

To handle this high dimensionality and the challenge of partial observability, we extend the framework of guided policy search. In guided policy search, the policy is trained with supervised learning, which scales well even to very high-dimensional function approximators. To construct the training set for supervised learning, we employ a trajectory-centric reinforcement learning method that finds good trajectories from a number of initial states. This phase does not require knowledge of the system dynamics, but does use the full state , which makes learning very efficient. For example, if the unobserved part of is the position of a target object, such as the bottle, we can hold this object in the robot's left gripper, while the right arm performs the task. This type of instrumented training is a natural fit for many robotic tasks,

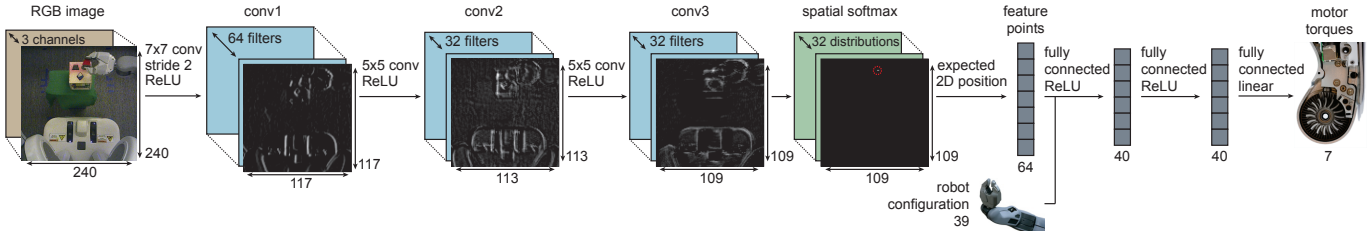


Fig. 2: Visuomotor policy architecture. The network contains three convolutional layers, followed by a spatial softmax and an expected position layer that converts pixel-wise features to feature points, which are better suited for spatial computations. The points are concatenated with the robot configuration, then passed through three fully connected layers to produce the torques.

where the training is performed in a controlled environment, but the final policy must be able to succeed "in the wild."

Naïve supervised learning will often fail to produce a good policy, since a small mistake on the part of the policy will put it in states that are not part of the training, causing compounding errors. To avoid this problem, the training data must come from the policy's own state distribution [**?**]. We use BADMM [**?**] to adapt the trajectories to the policy, alternating between optimizing the policy to match the trajectories, and optimizing the trajectories to minimize cost and match the policy, such that at convergence, they have the same state distribution.

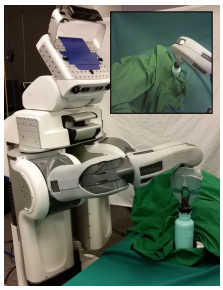
A. Network Architecture

Our visuomotor policy runs at 20 Hz on the robot, mapping monocular RGB images and the robot configurations to joint torques on a 7 DoF arm. The configuration includes the angles of the joints and the pose of the end-effector (defined by 3 points), as well as their velocities, but does not include the position of the target object or goal, which must be determined from the image. CNNs often use pooling to discard the locational information that is necessary to determine positions, since it is an irrelevant distractor for tasks such as object classification [**?**]. Because locational information is important for control, our policy does not use pooling. Additionally, CNNs built for spatial tasks such as human pose estimation often also rely on the availability of location labels in image-space, such as hand-labeled keypoints [**?**]. We propose a novel CNN architecture capable of estimating spatial information from an image without direct supervision in image space. Our pose estimation experiments, discussed in Section III-B, show that this network can learn useful visual features using only 3D positional information provided by the robot and no camera calibration. Furthermore, by training our network with guided policy search, it can acquire *task-specific* visual features that improve policy performance.

Our network architecture is shown in Figure 2. The visual processing layers of the network consist of three convolutional layers, each of which learns a bank of filters that are applied to patches centered on every pixel of its input. These filters form a hierarchy of local image features. Each convolutional layer is followed by a rectifying nonlinearity of the form $a_{cij} = max(0, z_{cij})$ for each channel c and each pixel coordinate (i, j). The third convolutional layer contains 32 response maps with resolution 109×109. These response maps are passed through a spatial softmax function of the form $s_{cij} = e^{a_{cij}} / \sum_{i'j'} e^{a_{ci'j'}}$. Each output channel of the softmax is a probability distribution over the location of a feature in the image. To convert from this distribution to a spatial representation, the network calculates the expected image position of each feature, yielding a 2D coordinate for each channel. These feature points are concatenated with the robot's configuration and fed through two fully connected layers, each with 40 rectified units, followed by linear connections to the torques. The full visuomotor policy contains about 92,000 parameters, of which 86,000 are in the convolutional layers.

The spatial softmax and the expected position computation serve to convert pixel-wise representations in the convolutional layers to spatial coordinate representations, which can be manipulated by the fully connected layers into 3D positions or motor torques. The softmax also provides lateral inhibition, which suppresses low, erroneous activations. This makes our policy more robust to distractors, providing generalization to novel visual variation. We compare our architecture with more standard alternatives in Section IV-C.

B. Training Data

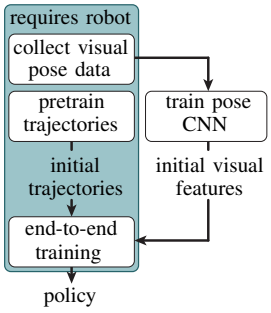


We train the policy using the full state during the trajectory optimization phase, though the final policy acts under partial observations. This type of instrumented training is a natural choice for many robotics tasks, where the robot is trained under controlled conditions, but must then act intelligently in uncontrolled, real-world situations. In our tasks, the unobserved variables are the pose of a target object (e.g. the bottle on which a cap must be placed). During training, this target object is held in the robot's left gripper, while the robot's right arm performs the task, as shown above. This allows the robot to move the target through a range of known positions. The final visuomotor policy does not receive this position as input, but must instead use the camera images. The left arm is covered with cloth to prevent the policy from associating its appearance with the object's position.

While we can train the visuomotor policy entirely on the robot, the algorithm would spend a large number of iterations learning basic visual features and arm motions that can more efficiently be learned by themselves, before being incorporated

into the policy. To speed up learning, we initialize both the vision layers in the policy and the trajectory distributions for guided policy search by leveraging the fully observed training setup. This initialization does not use any additional information that is not already available from the robot. To initialize the vision layers, the robot moves the target object through a range of random positions, recording the object's pose and camera images. This dataset is used to train a pose regression CNN, which consists of the same vision layers as the policy, followed by a fully connected layer that outputs the 3D points that define the target. Since the training set is still quite limited, we initialize the filters in the first layer with weights from the model of Szegedy et al. [**?**], which is trained on ImageNet [**?**] classification. After training on pose regression, the weights in the convolutional layers are transferred to the policy CNN. This enables the robot to learn the appearance of the objects prior to learning the behavior.

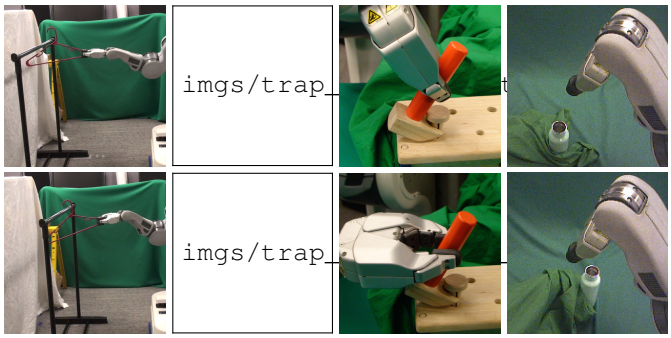
To initialize the trajectories, we take 15 iterations of guided policy search without optimizing the visuomotor policy. This allows for much faster training in the early iterations, when the trajectories are not yet successful, and optimizing the full visuomotor policy is unnecessarily time consuming. Since we still want the trajectories to arrive at compatible strategies for each target position, we replace the visuomotor policy during these iterations with a small network that receives the full state. This network serves only to constrain the trajectories and avoid divergent behaviors from emerging for similar initial states, which would make subsequent policy learning difficult. As shown in the diagram above, the trajectories can be pre-trained in parallel with the vision layer pre-training, which does not require the robot.



After initialization, we train the full visuomotor policy with guided policy search. During the supervised policy optimization phase, the fully connected motor control layers are first optimized by themselves, since they are not initialized with pre-training. Then, the entire network is further optimized end-to-end. We found that this setup prevents the convolutional layers from forgetting the useful features learned during pre-training while still training all layers of the policy.

IV. Experimental Results

We evaluated our method by training policies for hanging a coat hanger on a clothes rack, inserting a block into a shape sorting cube, fitting the claw of a toy hammer under a nail with various grasps, and screwing on a bottle cap. The cost function for these tasks encourages low distance between three points on the end-effector and corresponding target points, low torques, and, for the bottle task, spinning the wrist. The equations for these cost functions follow prior work [**?**]. The tasks are illustrated in Figure 3. Each task involved variation of about 10-20 cm in each direction in the position of the target object (the rack, shape sorting cube, nail, and bottle).



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 3: Illustration of the tasks in our experiments, showing the variation in the position of the target for the hanger, cube, and bottle tasks, as well as two of the three grasps for the hammer, which also included variation in position (not shown).

In addition, the coat hanger and hammer tasks were trained with two and three grasps, respectively. All tasks used the same policy architecture and model parameters.

A. Visuomotor Policy Generalization

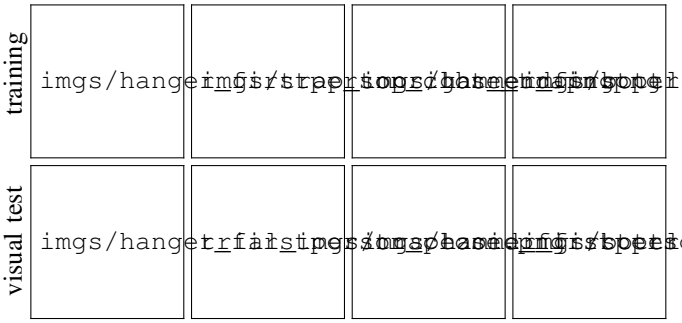
We evaluated the visuomotor policies in three conditions: (1) the training target positions and grasps, (2) new target positions not seen during training and, for the hammer, new grasps (spatial test), and (3) training positions with visual distractors (visual test). A selection of these experiments is shown in the supplementary video.[1] For the visual test, the shape sorting cube was placed on a table rather than held in the gripper, the coat hanger was placed on a rack with clothes, and the bottle and hammer tasks were done in the presence of clutter. Illustrations of this test are shown in Figure 4.

The success rates for each test are shown in Table I. We compared to two baselines, both of which train the vision layers in advance for pose prediction, instead of training the entire policy end-to-end. The features baseline discards the last layer of the pose predictor and uses the feature points, resulting in the same architecture as our policy, while the prediction baseline feeds the predicted pose into the control layers.

The pose prediction baseline is analogous to a standard modular approach to policy learning, where the vision system is first trained to localize the target, and the policy is trained on top of it. This variant achieves poor performance, because although the pose is accurate to about 1 cm, this is insufficient for such precise tasks. As shown in the video, the shape sorting cube and bottle cap insertions have tolerances of just a few millimeters. Such accuracy is difficult to achieve even with calibrated cameras and checkerboards. Indeed, prior work has reported that the PR2 can maintain a camera to end effector accuracy of about 2 cm during open loop motion [**?**]. This suggests that the failure of this baseline is not atypical, and that our visuomotor policies are learning visual features and control strategies that improve the robot's accuracy.

When provided with pose estimation features, the policy has more freedom in how it uses the visual information, and

[1]The video can be viewed at http://sites.google.com/site/visuomotorpolicy



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 4: Training and visual test scenes as seen by the policy at the ends of successful episodes. The hammer and bottle images were cropped for visualization only.

(a) conv3 (b) softmax (c) feature points

Fig. 5: Feature points learned by the shape sorting cube policy. Two of the 32 conv3 response maps are shown in (a), and the corresponding softmax distributions are displayed in (b). In (c), we show the output feature points for this input image in blue, while the feature points of the pose prediction network are shown in red. The end-to-end trained model discovers more feature points on the cube and the gripper.

achieves somewhat higher success rates. However, full end-to-end training performs significantly better, achieving high accuracy even on the challenging bottle task, and successfully adapting to the variety of grasps on the hammer task. This suggests that, although the vision layer pre-training is clearly beneficial for reducing computation time, it is not sufficient by itself for discovering good features for visuomotor policies.

coat hanger	training (18)	spatial test (24)	visual test (18)
end-to-end training	**100%**	**100%**	**100%**
pose features	88.9%	87.5%	83.3%
pose prediction	55.6%	58.3%	66.7%
shape sorting cube	training (27)	spatial test (36)	visual test (40)
end-to-end training	**96.3%**	**91.7%**	**87.5%**
pose features	70.4%	83.3%	40%
pose prediction	0%	0%	n/a
toy claw hammer	training (45)	spatial test (60)	visual test (60)
end-to-end training	**91.1%**	**86.7%**	**78.3%**
pose features	62.2%	75.0%	53.3%
pose prediction	8.9%	18.3%	n/a
bottle cap	training (27)	spatial test (12)	visual test (40)
end-to-end training	**88.9%**	**83.3%**	**62.5%**
pose features	55.6%	58.3%	27.5%

TABLE I: Success rates on training positions, on novel test positions, and in the presence of visual distractors. The number of trials per test is shown in parentheses.

The policies exhibit moderate tolerance to distractors that are visually separated from the target object. However, as expected, they tend to perform poorly under drastic changes to the backdrop, or when the distractors are adjacent to or occluding the manipulated objects, as shown in the supplementary video. In future work, this could be mitigated by varying the scene at training time, or by artificially augmenting the image samples with synthetic transformations, as discussed in prior work in computer vision [**?**].

B. Features Learned with End-to-End Training

In Figure 5, we compare the feature points learned through guided policy search to those learned by a CNN trained for pose prediction. After end-to-end training, the policy acquired a distinctly different set of feature points compared to the pose prediction CNN used for initialization. The end-to-end trained model finds more feature points on task-relevant objects and fewer points on background objects. This suggests that the policy improves its performance by acquiring *task-specific* visual features that differ from those learned for object localization. We further analyze the features learned by our policies in the supplementary appendix.

C. CNN Architecture Evaluation

To evaluate the visual processing portion of our architecture, we measured its accuracy on the pose estimation pre-training task discussed in Section III-B. We compare to a network where the fixed transformation from the softmax to the feature points is replaced with a conventional learned fully connected layer, as well as to networks that omit the softmax and use 3×3 max pooling with stride 2 at the first two layers. These alternative architectures have many more parameters, since the new fully connected layer takes as input the entire bank of response maps from the third convolutional layer. The results in Table II indicate that using the softmax and the fixed transformation from the softmax output to the spatial feature representation improves pose estimation accuracy and reduces overfitting. Our network is able to outperform the more standard architectures because it is forced by the softmax and expected position layers to learn feature points, which provide a concise representation suitable for spatial inference. The lower number of parameters also results in an easier optimization and reduces overfitting.

D. Implementation and Computational Performance

CNN training was implemented using the Caffe [**?**] deep learning library. Each visuomotor policy required 3-4 hours of training time: 20-30 minutes for the pose prediction data collection on the robot, 40-60 minutes for the fully observed trajectory pre-training on the robot and offline pose pre-training (which can be done in parallel), and between 1.5

network architecture	training error (cm)	test error (cm)
softmax + feature points (**ours**)	**1.14 ± 1.67**	**1.30 ± 0.73**
softmax + fully connected layer	2.27 ± 1.70	2.59 ± 1.19
fully connected layer	4.65 ± 2.90	4.75 ± 2.29
max-pooling + fully connected	2.89 ± 2.08	3.71 ± 1.73

TABLE II: Average pose estimation accuracy and standard deviation with various architectures, measured as average Euclidean error for the three target points in 3D, with ground truth determined by forward kinematics from the left arm.

and 2.5 hours for end-to-end training with guided policy search. The coat hanger task required two iterations of guided policy search, the shape sorting cube and the hammer required three, and the bottle task required four. Training time was dominated by computation rather than robot interaction time, and we expect significant speedup from a more efficient implementation.

V. Discussion and Future Work

In this paper, we presented a method for learning robotic control policies that use raw input from a monocular camera. These policies are represented by a novel convolutional neural network architecture, and can be trained end-to-end using our partially observed guided policy search algorithm, which decomposes the policy search problem in a trajectory optimization phase that uses full state information and a supervised learning phase that only uses partial observations. This decomposition allows us to leverage state-of-the-art tools from supervised learning, making it straightforward to optimize extremely high-dimensional policies. Our experimental results show that our method can execute complex manipulation skills, and that end-to-end training produces significant improvements in policy performance compared to using fixed vision layers trained for pose prediction.

Although we demonstrate moderate generalization over variations in the scene, our current method does not generalize to dramatically different settings, especially when visual distractors occlude the manipulated object or break up its silhouette in ways that differ from the training. The success of CNNs on exceedingly challenging vision tasks suggests that this class of models is capable of learning invariance to irrelevant distractor features [**? ? ?**], and in principle this issue can be addressed by training the policy in a variety of environments, though this poses certain logistical challenges. More practical alternatives that could be explored in future work include simultaneously training the policy on multiple robots, each of which is located in a different environment, developing more sophisticated regularization and pre-training techniques to avoid overfitting, and introducing artificial data augmentation to encourage the policy to be invariant to irrelevant clutter. However, even without these improvements, our method has numerous applications in, for example, an industrial setting where the robot must repeatedly and efficiently perform a task that requires visual feedback under moderate variation in background and clutter conditions.

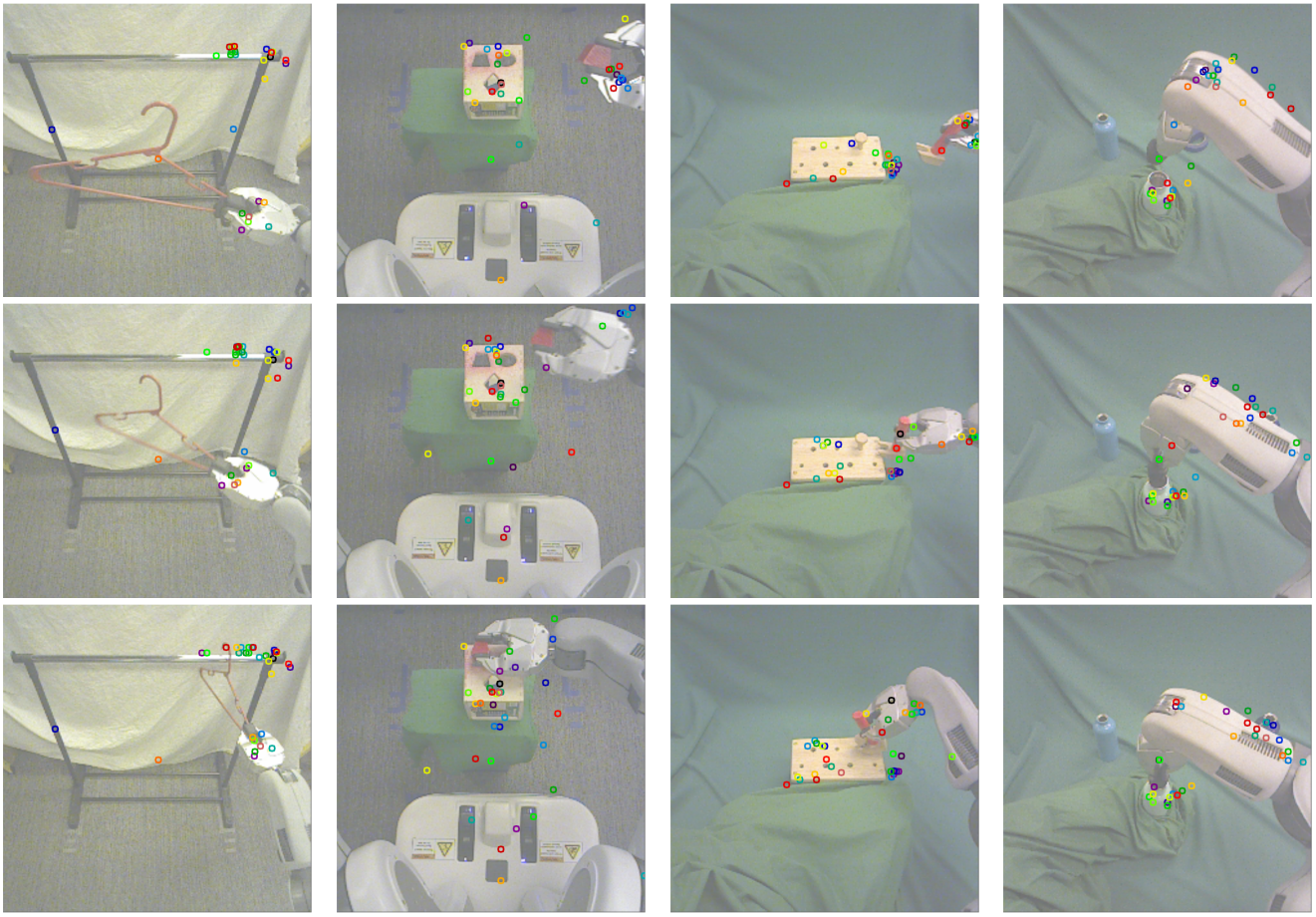
In future work, we hope to explore more complex policy architectures, such as recurrent policies that can deal with extensive occlusions by keeping a memory of past observations. We also hope to extend our method to a wider range of tasks that can benefit from visual input, as well as a variety of other rich sensory modalities, including haptic input from pressure sensors and auditory input. With a wider range of sensory modalities, end-to-end training of sensorimotor policies will become increasingly important: while it is often straightforward to imagine how vision might help to localize the position of an object in the scene, it is much less apparent how sound can be integrated into robotic control. A learned sensorimotor policy would be able to naturally integrate a wide range of modalities and utilize them to directly aid in control.

A. Feature Point Analysis

The visual processing layers of our architecture automatically learn features points using the fixed transformation from the softmax to spatial coordinates. These feature points encapsulate all of the visual information received by the motor layers of the policy. In Figure 6, we show the features points discovered by our visuomotor policy through guided policy search. Each policy learns features on the target object and the robot manipulator, both clearly relevant to task execution. The policy tends to pick out robust, distinctive features on the objects, such as the left pole of the clothes rack, the left corners of the shape-sorting cube, the bottom-left corner of the toy tool bench, and the edges of the bottle.

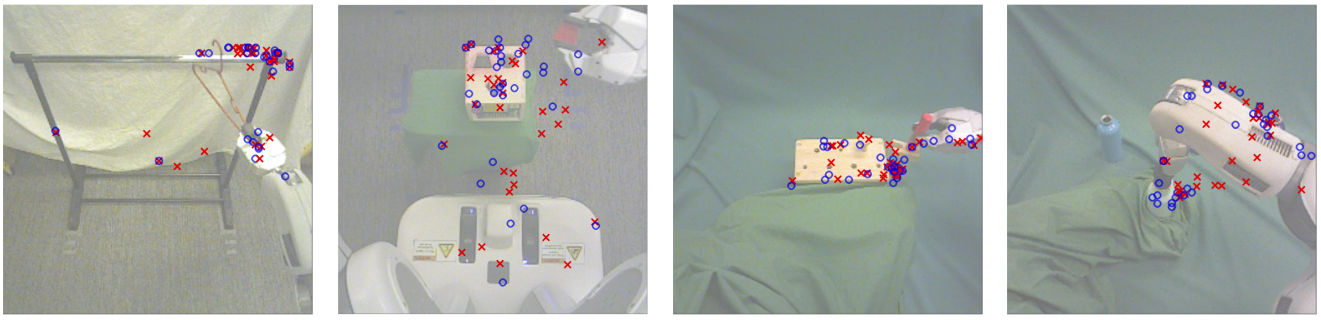
In Figure 7, we compare the feature points learned through guided policy search to those learned by a CNN trained for pose prediction. Note that in all tasks, the end-to-end trained model produces fewer points on the background compared to the model trained on object pose. In the bottle task, the end-to-end trained policy outputs points on both sides of the bottle, including one on the cap, while the pose prediction network only finds points on the right edge of the bottle.

The feature point representation is very simple, since it assumes that the learned features are present at all times. While this is a drastic simplification, both the pose predictor and the policy still achieve good results. A more flexible architecture that still learns a concise feature point representation could further improve policy performance. We hope to explore this in future work.



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 6: Feature points tracked by the policy during task execution for each of the four tasks. Each feature point is displayed in a different random color, with consistent coloring across images. The policy finds features on the target object and the robot gripper and arm. In the bottle cap task, note that the policy correctly ignores the distractor bottle in the background, even though it was not present during training.



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 7: Feature points learned for each task. For each input image, the feature points produced by the policy are shown in blue, while the feature points of the pose prediction network are shown in red. The end-to-end trained policy tends to discover more feature points on the target object and the robot arm than the pose prediction network.