Part III

Learning approach – Towards learning PPS through robot actions
Chapter 6

Learning Visuomotor mapping and Transferring learning from Simulation to the Real World

6.1 Introduction

Humans are able to achieve remarkably accurate and fast hand-eye coordination when performing tasks such as reaching and grasping. A key requirement for this skill is the association of a visual perception with proprioception by means of visuomotor mapping [Johansson et al., 2001]. It has been suggested that infants develop this visuomotor mapping during the development of reaching skills [Bushnell, 1985, Corbetta et al., 2014].

Most previous works in robotics tackle the visuomotor mapping in two steps [Hollerbach et al., 2008]. The first step is to obtain the robot’s kinematic model, which makes it possible to find the position of a joint in Cartesian space given the joint configuration (i.e. the joint angles). If a physical model of the robot is available (e.g. a CAD model) the kinematic model can be represented in the Denavit–Hartenberg notation. An alternative is to learn the kinematic model without using a priori information, as exemplified in [Zambelli and Demiris, 2017, Antonelli et al., 2011].

The second step, called visual-based pose estimation, serves to improve the robustness and the accuracy of the first step and consists of finding the pose of the end-effector within the images acquired from the robot cameras. This is usually performed from a single image or a stereo pair of images [Hollerbach et al., 2008]. In the stereo vision case, visual features are extracted to find the position of the end-effector in 2D space, and the disparity between the left and right images is used to find the 3D position with respect to the reference frame of the camera. Using the kinematic model obtained in the first step, this position can then be expressed in the robot’s root reference frame.
The two-step approach outlined above has several drawbacks, however. As highlighted in [Kim et al., 2013], there are various sources of error involved in both steps. Within the first step, even if an accurate kinematic model exists, precise position control is hard to achieve using a cable driven robot like the iCub. Furthermore, some mechanical pieces such as the iCub eyes need to be re-calibrated at each startup of the robot. Similarly, the visual-based pose estimation step is often inaccurate since the appearance of the end-effector in an image is highly dependent on the head pose, which itself has inherent uncertainties for the same reasons outlined above. The visual-based pose estimation step also relies on precise estimates of the camera parameters, which is tackled elsewhere [Fanello et al., 2014]. To minimize the effect of these inaccuracies, an additional calibration step has been suggested to find a set of offsets specifically for hand-eye coordination [Horaud and Dornaika, 1995, Hager et al., 1995, Hollerbach et al., 2008, Fanello et al., 2014].

Here it is proposed to find the visuomotor mapping in a single step, rather than considering the two problems independently and then subsequently finding an offset mapping, as outlined above. More specifically, and as shown in Fig. 6.1, I suggest to learn the mapping from an imprecise model in simulation using two components: 1) A deep neural network estimates...
6.2. Related Works

Learning a visuomotor mapping: Some methods closely follow the classical approach for visuomotor mapping outlined in the introduction, but use machine learning methods to make the mapping more robust and adaptable. For example, Antonelli et al. [2013] rely on several radial basis functions, although their method is limited in the sense that markers are required for feature extraction and the disparity is assumed to be known.

To find the end-effector pose using visual information, it has been suggested to compare the hand perceived by the robot’s camera with a realistically rendered hand from simulation [Vicente et al., 2016, Fantacci et al., 2017]. A particle filter is then used to predict the 6D pose of the robot’s hand, which is used for a visual servoing reaching task.

The hand-eye coordination task has also been investigated within the developmental robotics domain. Aguilar and Pérez [2017] propose a method which allows for coordination of the visual and tactile modalities, i.e. the robot’s hand is equipped with tactile sensors which can perceive the presence of objects using touch. The focus of this method is on the emergence of higher-level behaviours, such as the exploration of various objects and following objects using attentional processes. Similarly, Hwang et al. [2017] investigate the emergence of a mirror neuron system for imitation learning using such a developmental approach. While Hwang et al. [2017] is limited to imitation learning using the same robot, Chang et al. [2016, 2017] present
Chapter 6. Learning Visuomotor mapping and Transferring learning from Simulation to the Real World

a method that makes it possible to find body shape correspondence between a number of robots and humans in images. This method relies on motion information, however, and is not suited to finding image matches where the bodies are in the same pose.

**Simulation to real transfer learning:** One of the aims of this work is to transfer skills from simulation to real robots, and several previous works have shown that this approach is viable. Rusu et al. [2017] focus on a reaching task using reinforcement learning. They use a concept called progressive networks that trains parts of the network in simulation and then using these parts to bootstrap another network that is applied on the physical robot. Interestingly, the method learns directly from raw visual input, and is capable of using the additional input modalities of the real robot that are not present in the simulator. Tzeng et al. [2015], meanwhile, propose using a domain confusion loss for end-effector pose estimation. The idea is to enforce the same feature representation regardless of whether the input is from simulation or the real robot. This is implemented using a discriminator that attempts to classify whether the feature originated from a simulated or real image.

Several works tackle the problem that the control policy learned on simulation data does not directly map to the real robot. Christiano et al. [2016] use a deep inverse model trained on the behaviour of the real robot to find actions that resembles the high-level properties of the policy found in simulation, while abstracting away the low-level properties that differ between the simulation and the real environment. This approach was extended by Sadeghi et al. [2018], who focus on finding a viewpoint-invariant control policy. Zhang et al. [2017] propose a modular deep network for robot reaching. A perception module estimates the target object location, which is then used by a control module that issues velocity commands to reach that target. They improve hand-eye coordination accuracy by end-to-end fine-tuning of these modules.

A completely different approach is taken by Tobin et al. [2017] and James et al. [2017], who propose that generalization abilities can be achieved by alterations of the rendered images of the simulator in terms of textures, lighting conditions, camera position and other factors. A distinct property of these methods is that they do not require any fine-tuning on real images. This is in stark contrast to [Zhang et al., 2018], where it is proposed to map the real world images back to the simulator domain rather than the more common method of finding a mapping into the real domain.

The work most similar to this one is that of [Bousmalis et al., 2018]. Like
the method used here, a generator is trained to produce realistic images from synthetic images, and a discriminator is trained to distinguish synthetically created images from real images. Their proposal is constrained to the grasping task, however, since an estimate of the grasping success is used as semantic input when training the generator. Since only the grasping success is considered, this relaxes the requirement to estimate the state of each joint accurately. The method proposed here is complementary to these task specific methods, since it aims to sidestep the manual calibration that is required in complex robots like the iCub, while being agnostic to the specific manipulation task.

6.3 Methodology

This section first present the Imperial-sim2real dataset, which contains images obtained both from the simulation and from the physical robot (Section 6.3.1). Then, an image-to-image translation method is introduced that maps images from the simulation domain to the real domain (Section 6.3.2). Finally, it is shown that these realistic looking images can be used to learn the visuomotor mapping (Section 6.3.3). All code is available on the GitHub repository.

6.3.1 Action-based dataset generation

The iCub humanoid robot (see Section 2.3) was utilized to create the Imperial-sim2real dataset. This chapter focuses on the seven DoFs of each arm and the six DoFs of the iCub’s head. Both eyes are equipped with a RGB camera with stereo vision capabilities [Fanello et al., 2014].

Dataset overview: The Imperial-sim2real dataset\(^2\) contains the following elements:

1. Sim: Stereo vision image pairs of the robot’s arm in simulation, along with the corresponding head joint and arm joint configurations.

2. Background: Background images collected using the physical robot without the robot arm in the visual field of view, with corresponding head joint configuration.

\(^1\)https://github.com/robotology/visuomotor-learning
\(^2\)https://doi.org/10.5281/zenodo.1186943
3. **TrainA**: Images created by combining *Sim* images and *Background* images of similar head configurations. Here the *Background* image with the most similar head configuration to the *Sim* image is used.

4. **TrainB**: Stereo vision image pairs of the physical robot’s arm.

5. **sim2real**: Synthetic images that were translated from the simulation domain into the real domain using an image-to-image translation method. Specifically, *TrainA* images are translated using CycleGAN, as described in Section 6.3.2.

**Motor babbling for randomized actions**: Since the purpose is to map the visual and motor spaces from one shot, the intention was to acquire data from the corresponding sensor sources (stereo vision image pairs, head configuration and arm configuration) in the whole working space of the robot’s arm using a motor babbling scheme.

More precisely, the working space was covered by issuing random actions to the relevant joints of the kinematic chain (i.e. from the eyes to the robot’s hand) and then storing the resulting stereo image pair, as well as the measurements of the joint encoders (seven arm joints and six head joints). For simplicity, the method was applied only to find the visuomotor mapping for the right hand, and thus the left hand was out of the field of view (although, of course, the mapping for the left hand can be found in the same way). The arm motions were generated using a velocity controller, following the proposal of [Zambelli and Demiris, 2017], as shown in Eq. (6.1), although their method was extended to also move the head joints using a position controller, as shown in Eq. (6.3). The superscripts $a$ and $h$ were utilized for arm and head respectively.

**Arm velocity controller**: The velocity command $\mathbf{v}^a$ was found as follows:

$$
\mathbf{v}^a = K \cdot (\mathbf{p}^a_{\text{ref}} - \mathbf{p}^a_t),
$$

(6.1)

with $\mathbf{p}^a_t$ being a vector of values containing the measurements of arm joints at time $t$, $K$ being the proportional gain. The reference values $\mathbf{p}^a_{\text{ref}}$ of the arm joints were generated by:

$$
\mathbf{p}^a_{\text{ref}} = \mathbf{p}^a_0 + \tilde{A} \cdot \sin(2\pi \tilde{f} t) \cdot \mathbf{1},
$$

(6.2)

where $\mathbf{1}$ duplicates a scalar into a vector of the appropriate size. The initial guess $\mathbf{p}^a_0$ ensures that the right arm remained visible throughout the
whole motion sequence, and \( \tilde{A} \sim \mathcal{N}(0, A) \) and \( \tilde{f} \sim \mathcal{N}(0, f) \) are normally distributed parameters for the amplitude and frequency, respectively.

**Head position controller:** The head position \( p^h \) was determined according to

\[
p^h = p^h_0 + \tilde{H} \cdot 1,
\]

(6.3)

where \( p^h_0 \) denotes the random initial head configuration and \( \tilde{H} \sim \mathcal{N}(0, H) \) is the normally distributed gain of the position controller. For safety reasons, the final control value sent to each joint was constrained by the firmware’s bounding value.

**Workspace coverage:** Covering a sufficiently large working space requires setting large values for \( A, H \) and \( f \). While this is feasible in simulation, lower values have to be used on the real robot due to mechanical stress constraints. For this work, the parameters were set as follows: \( A = 5, f = 0.2 \) and \( H = 10 \) in simulation or \( H = 5 \) for the physical robot. Thus, using the iCub simulator [Tikhanoff et al., 2008] has the additional benefit that it allows the collection of a larger, more diverse dataset compared to the physical robot. The resulting arm coverage in the simulated workspace is shown in Fig. 6.2, with the arm’s sweeping volume approximated as \( V \approx S_{\text{convex}} \cdot \bar{d} = 0.39 \text{m}^3 \) (where \( S_{\text{convex}} \) is the area covered by the convex hull spanned in the plane of the y- and z-axes, and \( \bar{d} \) is the average distance between the robot’s end-effector and the robot’s shoulder in the direction of the x-axis [motion in the x-axis is negligible]).

### 6.3.2 Image-to-image translation from the simulator to the real domain

This section proposes a method to learn a mapping function that maps images containing the iCub’s arm from the simulation domain to the real domain. In the computer vision domain, image-to-image translation has been addressed as learning a mapping function between an input image \( \{a_i\}_{i=1}^N \in \text{TrainA} \) and an output image \( \{b_i\}_{i=1}^N \in \text{TrainB} \) by training aligned image pairs [Efros and Leung, 1999]. The following will abbreviate \( \text{TrainA} \) to \( A \) and \( \text{TrainB} \) to \( B \) for brevity. In this task there were only two independent sets of images, with one consisting of simulation arm images \( A \) and the other consisting of real robot arm images \( B \) – thus there is no paired data indicating how a simulated image could be translated to a corresponding real image.
Chapter 6. Learning Visuomotor mapping and Transferring learning from Simulation to the Real World

In fact, as outlined in the introduction, one cannot rely on the joint configuration obtained by the physical robot since there are various unavoidable sources of errors.

Recently, generative adversarial network (GAN) [Goodfellow et al., 2014] based methods have shown good performance in generating realistic images, and some variants utilizing GANs have been applied to the image translation task without requiring aligned image pairs. This work adopts a state-of-the-art image-to-image translation method called Cycle-Consistent Adversarial Network (CycleGAN) [Zhu et al., 2017]. The CycleGAN is based on a combination of adversarial loss \( \mathcal{L}_{GAN} \) and cycle consistency loss \( \mathcal{L}_{cyc} \) in order to learn two mapping functions \( G_B : A \rightarrow B \) and \( G_A : B \rightarrow A \). The following section briefly describes the architecture and implementation of CycleGAN in my framework (see the Fig. 6.3 for descriptive diagram).

First, two adversarial discriminative networks (Discriminator) \( D_A \) and
6.3. Methodology

$D_B$ were introduced. $D_A$ distinguishes between real images $\{a\}$ and translated images $\{G_A(b)\}$ and similarly $D_B$ distinguishes between images $\{b\}$ and $\{G_B(a)\}$. The generators $G_A$ and $G_B$ tried to generate images $G_A(b)$ and $G_B(a)$ that look similar to images from the $A$ and $B$ domains, respectively. The adversarial loss for the mapping $G_B : A \rightarrow B$ is

$$L_{GAN}(G_B, D_B, A, B) = \mathbb{E}_{a \sim P_a} \left[ \log \left( 1 - D_B(G_B(a)) \right) \right] + \mathbb{E}_{b \sim P_b} \left[ \log D_B(b) \right] ,$$

and the mapping $G_A : B \rightarrow A$ is similarly designed as $L_{GAN}(G_A, D_A, B, A)$.

Second, the cycle consistency loss was designed to regularize the mappings, i.e. translate from one domain to the other, by ensuring that translating an already translated image back to the original domain should resemble the original image: $a \rightarrow G_B(a) \rightarrow G_A(G_B(a)) \approx a$. The cycle consistency loss is formulated as

$$L_{cyc}(G_B, G_A) = \mathbb{E}_{a \sim P_a} \left[ \| G_A(G_B(a)) - a \|_1 \right] + \mathbb{E}_{b \sim P_b} \left[ \| G_B(G_A(b)) - b \|_1 \right] ,$$
where $\| \cdot \|_1$ indicates the $l_1$-norm. By merging the adversarial losses and cycle consistency loss, the full loss becomes

$$L(G_A, G_B, D_A, D_B) = L_{GAN}(G_B, D_B, A, B) + L_{GAN}(G_A, D_A, B, A) + \lambda L_{cyc}(G_B, G_A),$$

(6.6)

where $\lambda$ controls the relative weight of the adversarial losses and the cycle consistency loss.

It is worth noting the importance of finding experimentally the correct value for $\lambda$ in the setting. If $\lambda$ is set too high, the CycleGAN tends to remember the learned images rather than generating new ones (in terms of interpolation and extrapolation). On the other hand, if $\lambda$ is set too low, the arm pose is not closely resembled in the synthesized images. It was experimentally found that $\lambda = 10$ represents a good compromise.

The model is implemented in Tensorflow and trained with the Adam optimizer [Kingma and Ba, 2015] using a learning rate of 0.0002 for both generators and discriminators.

### 6.3.3 Learning the visuomotor mapping

The visuomotor learning is treated as a regression problem, where, given a pair of images containing the robot’s arm (from the stereo-vision system) and the head joint configuration of the robot (neck-eyes), the aim is to estimate the corresponding joint configuration of the arm. An input sample ($x \in X$), therefore, contains two images and six measurements of the head joints, while a corresponding output sample ($y \in Y$) contains seven measurements of the robot’s arm joints. It is assumed that there is only one of the robot’s two arms contained in the input samples. The regression problem is then to approximate the function $f$ to map the input and output, i.e. $Y = f(X)$.

To estimate the mapping function $f$, a DNN model is proposed, as depicted in Fig. 6.4. It contains the following components:

- Two identical VGG19 [Simonyan and Zisserman, 2015] networks (see Fig. 6.5 for the details of the network architecture) to extract features from images obtained by both the left and right eye cameras. The feature vector of each VGG19 network was first routed to a dense layer of 1024 fully connected units, followed by a batch normalization layer and a ReLU activation layer. The dense layers were then concatenated and
integrated by a dense layer of 512 units. This deep visual output can be considered as an implicit 3D estimate from the raw stereo image pair.

- The head joint configuration was concatenated to this deep visual output, and then further processed in a densely connected network composed of three layers with 512, 256 and 7 units respectively. Finally, the predicted joint values $\hat{Y}$ were obtained after applying a $\tanh$ activation layer.

In contrast to the classical approach, which maps a visually perceived object to Cartesian space, and then from Cartesian space to the motor space, the proposed mapping $f$ directly infers the configuration of the arm joints in motor space from the spatial relation of pixels in raw images. Thus, this method does not suffer from the infinite solution problem of the inverse kinematics transformation (in redundancy manipulators) as occurs in the last step of the classical approach. In other words, the method considers a robot’s arm in the visual space as a whole rather than as a single, representative point in Cartesian space (also called the tool centre point).
The training process consisted of two steps. First, to bootstrap the network, only the Sim dataset which contains the unmodified simulator images was used. This results in the sim predictor. This network was then further fine-tuned using samples from the sim2real dataset (obtained using Cycle-GAN), resulting in the real predictor. This training scheme was chosen since the sim2real dataset is significantly smaller compared to the Sim dataset (some images are discarded in the sim2real dataset, as described in Section 6.4.2).

The DNN model was implemented in Keras\(^3\) with Tensorflow back-end, and was trained with the mean squared error loss function using the Adam optimizer [Kingma and Ba, 2015]. A batch size of 160 and a learning rate of 0.0001 were used.

### 6.4 Experiments and Results

This section evaluates the proposed method in a number of settings. The first experiment purely evaluates the DNN used for the visuomotor mapping (Section 6.4.1). The second experiment evaluates the image-to-image translation method and shows that realistic images can be generated from simulation while maintaining the semantic information of the arm pose (Section 6.4.2). The third experiment is a combination of the first two, and evaluates the visuomotor mapping in the real domain, i.e. using the physical robot and

\(^3\)https://keras.io/
the sim2real dataset (Section 6.4.3). Finally, I show that a simple calibrator using the offsets between the measured joint values and predicted joint values can be used in an object reaching scenario (Section 6.4.4).

6.4.1 Robot hand visual tracking in simulation

The first experiment evaluated the visuomotor DNN that was presented in Section 6.3.3 in simulation. The model was trained with \( N = 34000 \) training samples obtained by the proposed motor babbling scheme. The training input consisted of a stereo image pair with the corresponding head configuration, and the network was trained to estimate the arm’s joint values.

The trained model was then tested on a testing set separate from the training and validation sets (containing 6000 samples). If the model is able to generalize to new joint configurations, the error between the measured joints of the simulator and the predicted joints should be small. Indeed, Fig. 6.6 confirms that this was the case since the average error was just \( 1.85 \pm 1.32 \) degrees.

This result also demonstrates that the proposed deep network can be used as a component within action planning without having to rely on inverse kinematics methods. For example, the network can be used to find the corresponding joint configuration of the robot’s arm touching an object. Knowing this, it is straightforward to issue motor commands to achieve this joint configuration.

Another advantage of the proposed visuomotor deep neural network is that the inference only takes \( \approx 10 \text{ms} \) (on a NVIDIA GTX 1080Ti), which is significantly faster than other approaches for hand-eye coordination on the iCub (e.g. > 100ms in [Vicente et al., 2016]).

6.4.2 Imperial-sim2real dataset

The synthetic images presented in this section were generated by the image-to-image translation method described in Section 6.3.2. The Sim element contained approximately 40000 training samples collected using the simulator, and there were over 8000 images in the Background and TrainA elements of the dataset. The TrainB element, which was collected using the physical robot, contained more than 4000 stereo image pairs. Originally 18000 stereo image pairs were collected, however it was found that using fewer pairs provided sufficient training data for the CycleGAN, while reducing training time and avoiding overfitting. The sim2real element contained approximately 9000
synthetic images. While theoretically it was possible to obtain as many synthetic images as there were in the Sim image set, in practice images that were too blurry, or where the synthetic arm was in an unrealistic pose, needed to be manually discarded. Fig. 6.7 shows some examples of images acquired using the iCub simulator (TrainA images), and their corresponding images in the real domain acquired using CycleGAN (sim2real images).

### 6.4.3 Robot hand visual tracking on the physical robot

The third experiment investigated whether the tracking also performs well on the real robot. Compared to the experiment in Section 6.4.1, the visuomotor deep network was trained with synthetic images obtained using CycleGAN as described in Section 6.3.2 (sim2real images). The network was then
6.4. Experiments and Results

**Figure 6.7:** Image-to-image translation results. Left: simulator images where the background was replaced with random background images of the real domain (TrainA set). Middle: corresponding realistic-looking images synthesized using CycleGAN (sim2real set). The pose of the synthesized arm closely resembles that of the simulator arm. Right: the real images that are most similar to the synthesized ones in the middle column (TrainB set).

tested on the physical robot, and Fig. 6.8 shows the error between the predicted and measured joint states. As expected, there were some discrepancies, which were hypothesised due to the imprecise calibration of the physical robot. Importantly, however, as shown in the next section, these discrepancies could be compensated by a simple calibrator constructed from the discrepancies.

### 6.4.4 Joint calibration on the physical robot

While this chapter’s main contribution has been to demonstrate the process of learning the visuomotor mapping, as described in Section 6.3, here the accuracy of that mapping is investigated using a simple calibrator. The idea is
Chapter 6. Learning Visuomotor mapping and Transferring learning from Simulation to the Real World

as follows: the motor babbling scheme (Section 6.3.1) was performed on the physical robot, and the images of the arm were recorded alongside the measured joint values $y$. Using the collected images and the real predictor (Section 6.3.3), it was possible to obtain (paired) predicted joint values $\hat{y}$. Then, the measured joint values were mapped to the predicted joint values using a simple polynomial. One example application shown in this section is object touching. Specifically, the polynomial can be used to map a target joint configuration $y^*$ obtained from inverse kinematics to the corrected joint configuration $\hat{y}^*$ that was used to touch the object (rather than using $y^*$ directly).

The calibrator was implemented as polynomial $\hat{y} = c_0 + c_1 \odot y + c_2 \odot y^2 + c_3 \odot y^3$, where $\hat{y}$ denotes the predicted joint values, $y$ denotes the measured joint values and $\odot$ denotes the Hadamard product. The $c_i$ are constant vectors that are approximated using the collected data in Section 6.4.3 by applying the least squares polynomial fitting method. This procedure can be fully

\footnote{I abuse notation and denote $y \odot y$ as $y^2$ and $y \odot y \odot y$ as $y^3$.}
automated and has to be repeated every time the mechanical properties of
the robot change, e.g. after a cable has been replaced.

This calibrator can then be used to ensure that the iCub accurately touches
objects in a table-top scenario. Using the stereo-vision and the eyes-to-root
transformation of the robot [Fanello et al., 2014], the Cartesian coordinates
(x,y,z) of a known object in the root reference frame were obtained. The
Cartesian coordinates were then converted to the corresponding target joint
configuration \( y^* \) using the inverse kinematic library of the robot (iKin,
Chapter 6. Learning Visuomotor mapping and Transferring learning from Simulation to the Real World

see [Pattacini, 2011]). It should be emphasized that the inverse kinematics library was only used to obtain the joint configuration corresponding to Cartesian coordinates, but not to control the robot. The calibrator was used simply to obtain the corrected joint configuration \( \hat{y}^* \) that was then issued to the robot.

The reaching accuracy was evaluated by varying the target object’s position and comparing the success with and without using the proposed calibration method. Fig. 6.9 shows that the robot failed to touch the object when using \( y^* \) directly (i.e. without using the proposed calibrator). When the corrected joint values \( \hat{y}^* \) were used, however, the robot could touch the object successfully. The supplementary video provides further details on this validation\(^5\).

6.5 Discussion

In this chapter, a new way of using the data obtained through a simulator in the real domain has been introduced. It has been shown that synthesized data can be used to predict the robot’s arm configuration given a pair of stereo images and knowledge of the head configuration. The discrepancies between these predictions and the measured arm joint values can be used to compute a simple calibrator, which is then used in a reaching task.

The next step will be applying the learned visuomotor mapping in vision-based action planning, including reaching with obstacle avoidance and object grasping. For the reaching with avoidance task, it is planned to use the proposed method to infer the arm joint configuration corresponding to the images of the robot’s arm that reaches or collide with objects. An advantage of the method demonstrated in this chapter is that the planning problem is mapped from vision space to joint space in real-time, where well-established motion planning techniques such as RRT* and PRM* [Karaman and Frazzoli, 2011] can be applied. Another benefit of the proposed framework is the feasibility of integrating the touch modality using the tactile sensors of the iCub.

One limitation of this method is that it is currently constrained to setups where the background closely resembles that of the dataset. Future work will seek to overcome this limitation using domain randomization techniques [Tobin et al., 2017, James et al., 2017]. Other future works include the simultaneous learning of the visuomotor mapping for both arms, extending the

\(^5\)https://youtu.be/VMw8sVztcKA
framework to learn directly from uncalibrated images, and simultaneously learning the joint state and end-effector position. Methods from the multi-task learning [Deisenroth et al., 2014] and transfer learning [Peng et al., 2018] fields will be explored in an effort to achieve these goals.