Understanding Multi-Modal Perception Using Behavioral Cloning for Peg-In-a-Hole Insertion Tasks

Romeres, Diego; Liu, Yifang; Jha, Devesh K.; Nikovski, Daniel N.

Abstract

One of the main challenges in peg-in-a-hole (PiH)insertion tasks is in handling the uncertainty in the location of the target hole. In order to address it, high-dimensional sensor inputs from sensor modalities such as vision, force/torque sensing, and proprioception can be combined to learn control policies that are robust to this uncertainty in the target pose. Whereas deep learning has shown success in recognizing objects and making decisions with high-dimensional inputs, the learning procedure might damage the robot when applying directly trialand-error algorithms on the real system. At the same time, learning from Demonstration (LfD) methods have been shown to achieve compelling performance in real robotic systems by leveraging demonstration data provided by experts. In this paper, we investigate the merits of multiple sensor modalities such as vision, force/torque sensors, and proprioception when combined to learn a controller for real world assembly operation tasks using LfD techniques. The study is limited to PiH insertions; we plan to extend the study to more experiments in the future. Additionally, we propose a multi-step-ahead loss function to improve the performance of the behavioral cloning method. Experimental results on a real manipulator support our findings, and show the effectiveness of the proposed loss function.

Understanding Multi-Modal Perception Using Behavioral Cloning for Peg-In-a-Hole Insertion Tasks

Yifang Liu
School of Electrical and Computer
Engineering, Cornell University
Ithaca, NY 14853
Email: yl892@cornell.edu

Diego Romeres
Mitsubishi Electric
Research Laboratories
Cambridge, MA 02139
Email: romeres@merl.com

Devesh K. Jha
Mitsubishi Electric
Research Laboratories
Cambridge, MA 02139
Email: jha@merl.com

Daniel Nikovski
Mitsubishi Electric
Research Laboratories
Cambridge, MA 02139
Email: nikovski@merl.com

Abstract—One of the main challenges in peg-in-a-hole (PiH) insertion tasks is in handling the uncertainty in the location of the target hole. In order to address it, high-dimensional sensor inputs from sensor modalities such as vision, force/torque sensing, and proprioception can be combined to learn control policies that are robust to this uncertainty in the target pose. Whereas deep learning has shown success in recognizing objects and making decisions with high-dimensional inputs, the learning procedure might damage the robot when applying directly trial-and-error algorithms on the real system. At the same time, learning from Demonstration (LfD) methods have been shown to achieve compelling performance in real robotic systems by leveraging demonstration data provided by experts. In this paper, we investigate the merits of multiple sensor modalities such as vision, force/torque sensors, and proprioception when combined to learn a controller for real world assembly operation tasks using LfD techniques. The study is limited to PiH insertions; we plan to extend the study to more experiments in the future. Additionally, we propose a multi-step-ahead loss function to improve the performance of the behavioral cloning method. Experimental results on a real manipulator support our findings, and show the effectiveness of the proposed loss function.
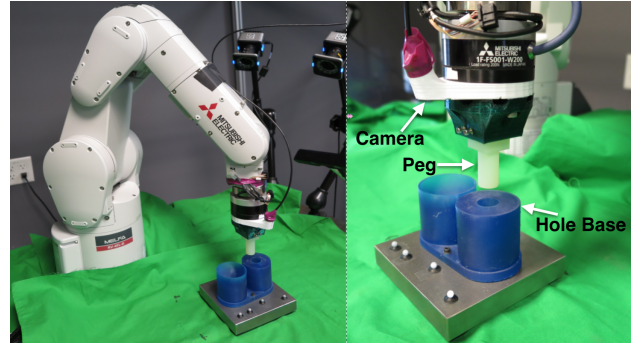
Fig. 1: Peg-in-a-hole experimental setup. Left: a Mitsubishi Electric RV-4FL model robot arm, and a 3D-printed cylindrical peg, a green peg holder, a white wrist camera holder (attached on the end-effector), and a blue hole base. Right: A closer view of the peg, the hole base, and the wrist camera (located underneath the purple tape with black wire).

I. INTRODUCTION

Contact-rich manipulation tasks, such as peg-in-a-hole (PiH) insertion, are very common in our daily lives, and can be performed effortlessly by humans. In contrast, it is still difficult for robots to perform these simple tasks when the target location is not specified accurately. If the target location is known exactly, methods such as motion planning and Dynamic Movement Primitives (DMP) [1, 2] can perform these tasks well, in combination with standard compliance controllers. However, these methods would typically fail if non-negligible uncertainty is present in the target location. Recently, vision-guided manipulation methods based on deep learning or deep reinforcement learning have received much attention, because they are robust to the variation of the target location [3, 4, 5]. However, these algorithms often require large amounts of training samples, good exploration strategies, or a carefully designed loss function due to the high dimensionality and non-linearity of the sensor inputs.

Learning from Demonstration (LfD) is a useful tool to speed up training, as it allows robots to leverage expert demonstration data to efficiently learn skills to perform manipulation tasks with acceptable performance. It can also be used to warm-start reinforcement learning techniques [6]. LfD

has been applied successfully to a wide range of domains in robotics, for example, in autonomous driving [7, 8] and manipulation [9]. Most LfD methods use humans as demonstrators [10]. However, human demonstrations are usually noisy and time-consuming. For tasks such as PiH, if the target location is known, which is usually the case at training time, data collection can become easier by implementing a basic controller on the robot instead of using human demonstrations. For these reasons, in this work, we use an automated controller to collect demonstrations. Behavioral cloning is an efficient imitation learning method, which learns the policy used by the expert during demonstration given the state of the environment [11, 12]. However, simple cloning models that are only trained on an expert's policy distribution are prone to mistakes due to the state distribution shift at test time, whereas more complex behavioral cloning methods that generalize to out-of-distribution states can be difficult to use [11]. We propose an efficient and simple behavioral cloning approach for PiH, based on a multi-step-ahead loss function. Whereas multi-step-ahead loss functions, as opposed to the more common one-step-ahead loss functions, have been already proposed for model learning [13, 14], to the best of our knowledge, they have never been proposed for model-free behavioral cloning algorithms. Their advantage is that, at each time step, the

policies trained with these loss functions enforce the system to end up in states that are still close to the demonstrations.

The main effort of this work is in combining and experimenting with different sensor modalities when performing insertions task in robotic systems. Indeed, various sensors such as cameras, force/torque (F/T) sensors, and positional sensors can (and should) be used both to provide feedback measurements as well as to give information about the target location when unknown. For instance, in PiH insertions, vision can provide information about the relative location of the hole w.r.t. the peg, and force-torque readings can monitor the forces exerted upon the peg allowing impedance and compliant control. Physics-based simulations, widely used in simulating body dynamics and collision detection, could also be a valid alternative to study the merits of different sensor modalities. However, the contact dynamics and friction properties during peg insertions are hard to model precisely in simulation. For this reason, understanding multi-modal perception on real robotic systems is still an open problem.

In practice, PiH tasks can be divided into two sub-tasks. The first is approaching the neighborhood of the hole with the peg, based on (possibly uncertain) information of the target location, and the second is the actual insertion into the hole. The former sub-task can be accomplished with standard trajectory planning techniques, or machine learning techniques such as DMP, when a demonstration is available. The approximate position of the hole can be known a priori or estimated by using motion-capture systems or off-the-shelf pose estimation techniques. As a result, the first sub-task can be solved relatively easily. The second sub-task is a more complex manipulation problem, where high precision control is required, and where predefined trajectories or open-loop controllers would most likely fail. Indeed, the controller needs to handle possible unforeseen impacts and contact forces. Understanding the role of different sensor modalities in real robotic systems is fundamental to successfully perform assembly tasks with significant uncertainty.

For these reasons, we are interested in solving the second sub-task. Focusing on a sub-task has the advantage of requiring less data, and avoids the use of computational power and learning capability to learn components such as the approach trajectory, that can be more efficiently computed in other ways.

A. Contributions

Motivated by the above discussion, we investigate methods for learning different controllers for a robotic manipulator to perform peg insertions starting from a neighborhood of the target hole. We rely on a behavioral cloning approach comparing deep-learning-based controllers designed to handle different input modalities. We focus on empirically analyzing the importance of different sensor modalities, i.e., images from a wrist camera, end-effector state information (proprioception), and force/torque measurements at the wrist of the robot. The first person point-of-view (p.o.v.) offers higher precision and every-day advantages such as simpler calibration procedures w.r.t. a third p.o.v. camera. Its use is also motivated by the

recent increase in the number of commercial robots equipped with cameras at the wrist. The main open questions we investigate are: 1) Is it always advantageous to consider all three sensor modalities? 2) Can we achieve success even if we do not specify the target location of the hole in spatial coordinates?

The main contributions of the paper can be summarized as:

- Empirical analysis of the effects of multi-modal perception in PiH insertions by combining several different sensor modalities into the control policy.
- Introduction of a multi-step-ahead loss function to improve the performance of the policy learned from demonstration data in a behavioral cloning approach.
- We obtained deep learning-based controllers that successfully perform PiH insertions and are goal independent at test time, which means that they can generalize over the robot workspace. Data collection and algorithm evaluation were done on a real robotic system.

II. RELATED WORK

One of the main challenges in contact-rich assembly tasks, such as peg insertion, stacking, and edge following, is that the operation should be robust to the uncertainty in the target location. The reason is that even a slight misalignment would create very large contact forces that might cause the task to fail or even damages the parts or the robot itself. Contact-rich assembly tasks, especially peg insertion, have been studied for a long time, due to their relevance in manufacturing. Previous work includes RL-based methods, such as [3], [15], [16], [17, 18, 4], LfD-based methods [19], [20], and combination of RL and LfD [5, 21, 22]. These previous papers use different combinations of sensing modalities. For instance, [15] studied the insertion of a rigid peg into a deformable hole, where the position of the hole relative to the robot is known, possibly with minor uncertainty. The authors combine information of the robot's state with filtered force/torque readings to form inputs to the controller. The controller trained in [19] uses RGB-D images as input, while position information was used in an auxiliary loss function during training. Furthermore, [5] proposed a DDPG-based approach. The authors used positions, velocities, torques, and visual features as observation input, and compared fixed, not completely fixed, and fully randomized goal locations. It was shown that the agent can solve the fixed position task without visual features, but such features were necessary for solving the task with randomized socket position. In [3], a study on the importance of three sensor modalities — RGB images, force readings, and proprioception — for peg insertion tasks was conducted in simulation.

LfD is a widely used technique in the machine learning community, usually involving a human (expert) as a demonstrator to provide demonstration about how to perform a task. For example, [19] applied Virtual Reality teleoperation to collect demonstrations, and behavior cloning was used to train visuomotor policies; [6] proposed deep Q-learning from demonstrations, which leverages small amounts of demonstration data to accelerate learning. The demonstration data is

collected by a human and a controller. Authors in [5] gathered demonstrations by using a 6D mouse (SpaceNavigator) and interpreted the motion as Cartesian commands, which they translated to joint actions using the known kinematics of the arm. In some other related work, human demonstrators directly guide the robot to finish the task by holding it. For example, [23] starts with a few correct human demonstrations obtained by a human holding the robot arm to perform the task, and then letting the robot learn a policy through trial and error. In [24], learning is initialized with two incorrect demonstrations by a human holding the robot. Learning proceeds through guided exploration around the demonstrations. Even though most existing methods use a human as a demonstrator, because industrial manipulators, like the one used in this paper, are not collaborative, and also due to the simplicity of obtaining demonstrations for this task, we make use of an automated controller to collect data.

III. LEARNING METHODS

In this section, we discuss the neural network architectures we use for training the multi-modal policies, and then describe the behavioral cloning method used in our experiments.

A. Neural Network-Based Controllers

We consider three different architectures of neural networks that all take as input raw observations, and provide as outputs the robot command actions. In particular, at each time step t, a network takes as input a subset of the following observation vector $o_t = (I_t, FT_t, ee_t)$ where $I_t \in \mathbb{R}^{60 \times 60 \times 3}$ is an RGB image from a general purpose web camera mounted on the wrist of the robot (first-person view), $FT_t \in \mathbb{R}^6$ is a force-torque sensor measurement from a wrist-mounted sensor, and $ee_t \in \mathbb{R}^3$ is the Cartesian position of the end-effector, which are proprioceptive measurements from the joint encoders built-in the robot arm. The outputs of all neural networks are the robot commands, defined as the desired change in position $\hat{\Delta}_t \in \mathbb{R}^3$ in robot base coordinate frame. The three types of neural networks considered are described in the following.

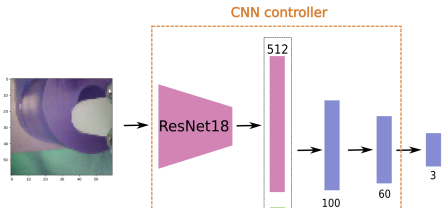*1) ResNet:* The neural network is illustrated in Fig. 2. We

Fig. 2: Architecture of the "ResNet" controller.

use a pre-trained ResNet18 [25] to extract a feature vector $f_t^I \in \mathbb{R}^{512}$ from the image input I_t. The features f_t^I are inputted into a network with 2 fully-connected layers that outputs the action vector Δ_t, see Fig. 2.
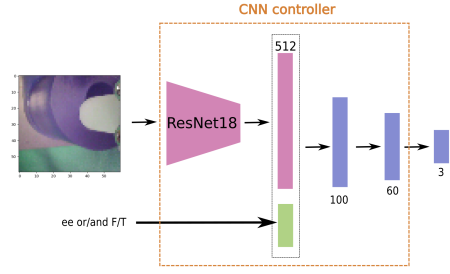

Fig. 3: Architecture of "ResNet + raw sensory data" controller.

2) ResNet + raw sensory data: The neural network is illustrated in Fig. 3. The same pre-trained ResNet18 [25] is used to extract the feature vector $f_t^I \in \mathbb{R}^{512}$ from the image input I_t. The FT_t and ee_t raw sensory data are concatenated with f_t^I into the feature vector f_t^{NN}, according to one of the following options that gives its name to several different control policies:

- **ResNet + ee raw.** Consider image I_t and ee_t as input. Then $f_t^{NN} \in \mathbb{R}^{515}$.
- **ResNet + FT raw.** Consider image I_t and FT_t as input. Then $f_t^{NN} \in \mathbb{R}^{518}$.
- **ResNet + ee + FT raw.** Consider image I_t, ee_t and FT_t as inputs. Then $f_t^{NN} \in \mathbb{R}^{521}$.

The feature vector f_t^{NN} is transformed into the action vector $\hat{\Delta}_t$ through a network with 2 fully-connected layers, see Fig. 3.

3) ResNet + features extraction: The neural network is illustrated in Fig. 4. The same pre-trained ResNet18 [25] is
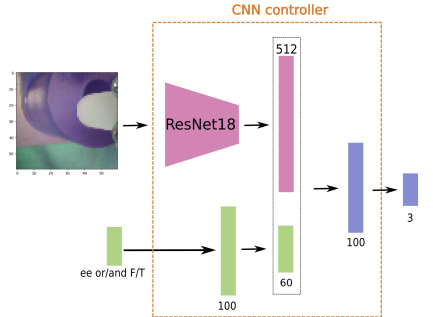

Fig. 4: Architecture of "ResNet+features extraction" controller

used to extract the feature vector $f_t^I \in \mathbb{R}^{512}$ from the image input I_t. The FT_t and ee_t raw sensory data are inputted into a fully connected network, which outputs a feature vector $f_t^r \in \mathbb{R}^{60}$, obtained by one of the following options that gives name to different control policies:

- **ResNet + ee features.** Input to the fully connected network is ee_t.
- **ResNet + FT features.** Input to the fully connected network is FT_t.
- **ResNet + ee + FT features.** Input to the fully connected network is the concatenation of ee_t and FT_t.

The feature vectors f_t^I and f_t^r are concatenated into the feature vector $f_t^{NN} \in \mathbb{R}^{572}$, and f_t^{NN} is transformed into the action vector $\hat{\Delta}_t$ with a fully connected network, see Fig. 4.

The orange dash-line boxes depicted in each of the Fig. 2,3,4 are denoted as the "CNN controller".

4) Other controllers: Controllers without the vision input might also be considered. All the combinations ee_t, FT_t and $ee_t + FT_t$ can be considered as inputs of a regression models or a neural network controller. For example we consider:
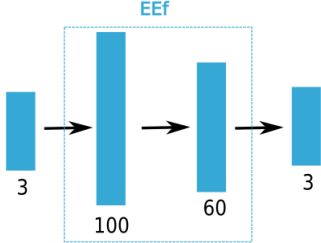

EEf

3   100   60   3

Fig. 5: Architecture of "EEf" controller.

EE features. The neural network is illustrated in Fig. 5.

The ee_t raw sensory data are inputted into a 2-layers fully connected network that outputs the action vector $\hat{\Delta}_t$. The blue dash-lined box in Fig. 5 is the network controller denoted by "EEf".

B. Multi-step-ahead Behavioral Cloning

Behavioral cloning is a supervised learning method for imitation learning and decision making. Using behavioral cloning, the robot might learn satisfactory control policies without knowing either the system dynamics or the cost function. However, it sometimes suffers from a state distribution shift issue. The Data Aggregation (DAgger) algorithm can improve performance, but it requires a continual presence of the demonstrators to respond to queries each time the robot encounters a new state [26]. To tackle the distribution mismatch problem without involving human supervisors, we propose to consider a multi-step-ahead loss function. The standard loss function used in behavioral cloning is the MSE of the action vector, used to match the demonstration of the expert as accurately as possible:

$$l = \frac{1}{T} \sum_{t=1}^{T} \|\hat{\Delta}_t - \Delta_t\|_2^2 \qquad (1)$$

where T is the number of training data examples.

We noticed that this loss function was not working well, because if the loss function was low in all the training points except in a few points the total loss function was still low, averaging out the loss in the critical points. In practice, during execution, the policy was not able to control the robot in some critical points, causing the experiment to fail. In order to incentivize policies that are accurate enough to control the robotic agent, we propose a loss function that considers multiple steps ahead, in open-loop fashion:

$$l = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{K} \sum_{k=1}^{K} \|\hat{\Delta}_{t+k} - \Delta_{t+k}\|_2^2 \qquad (2)$$

where K is the horizon of the multi-step-ahead prediction. In this way, the trained controller tends to guide the peg to an attraction area, which is within the distribution induced by the expert's demonstration. Consecutive predictions $\hat{\Delta}_{t+k}$ are obtained, as illustrated in Fig. 6. The 1-step-ahead prediction $\hat{\Delta}_{t+1}$ is obtained by one of the CNN controllers described in Sec. III-A. However, in order to predict the following actions $[\hat{\Delta}_{t+2}, \ldots, \hat{\Delta}_{t+K}]$, the CNN controller would require an estimate of the observations at the future time instants $[\hat{o}_{t+1}, \ldots, \hat{o}_{t+K-1}]$, which is a hard task (and, as we will see in the experiments, not really necessary). Instead, we propose to use the "EEf" controller introduced in Sec III-A, Fig.5. The prediction at $t + k$ steps ahead with $k > 1$ is computed as

$$\hat{ee}_{t+k} = \hat{ee}_{t+k-1} + EEf(\hat{ee}_{t+k-1}) = \hat{ee}_{t+k-1} + \hat{\Delta}_t$$

The multi-step-ahead model is shown in Fig. 6. The proposed loss function can also be used in other learning frameworks. More complicated loss functions than Eq. (1) have been taken in consideration, too, see e.g. [19]. Note that the multi-step-ahead loss functions could be applied to those loss functions, too.

IV. EXPERIMENTS

The aim of the experiments described below is to empirically evaluate the different controllers described in Section III based on different sensor modalities, and also investigate the effectiveness of the multi-step-ahead loss function proposed in Section III-B. In particular, this section will answer the following questions:

- Does the proposed multi-step-ahead loss function improve the training process compared to the traditional MSE loss function?
- Is it better to concatenate the image features with the raw end-effector and force/torque measurements, or with the features extracted from these raw measurements obtained as in "ResNet + features extraction"?
- Is it always beneficial to combine vision, force/torque readings, and end-effector positions into one controller?

To answer these questions, intensive experiments have been conducted on a real system. It is worth noticing that there is no comparison with other models, as the main goal of this paper is to understand different sensor modalities and evaluate if multi-step-ahead loss model improves the performance.

A. Experimental Setup

Fig. 1 shows the experimental setup. A MELFA RV-4FL robotic arm is used in the insertion task. The robotic arm is equipped with the following sensors: a Mitsubishi Electric 1F-FS001-W200 force/torque sensor attached to the wrist of the robot, a general-purpose RGB web camera mounted on top of the end-effector, and a set of built-in joint encoders that measure the robot's state. The experiments are conducted using custom 3D-printed components that include a cylindrical peg, a peg holder mounted on the wrist after the force/torque sensor, and a (blue cylindrical) box with a hole in the center.
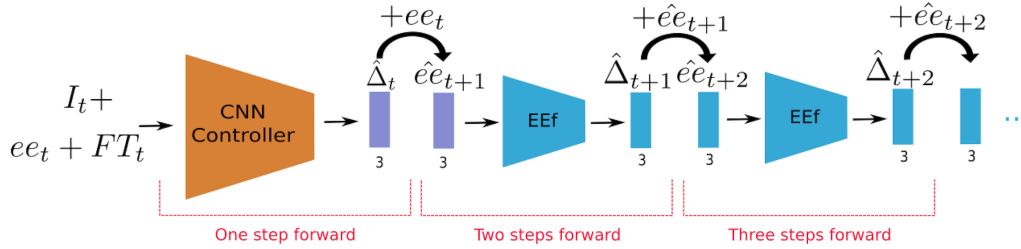
Fig. 6: Multi-step-ahead model. It shows the example with three steps. The "CNN Controller" is used to predict the first step ahead, then the second and third steps are predicted by the "Step NN" model.

The clearance between the peg and the hole is at the sub-millimeter level, measured at 0.7mm. The hole is attached to a heavy steel base to increase the stability of the hole's location during insertion. The robot is operated in position-control mode. During testing, the hole location is unknown, but is always in the field of view of the wrist camera.

B. Data Collection

An expert teacher needs to provide demonstrations. In PiH tasks, it is relatively easy to ascertain the location of the hole during training, so collecting data from an automated controller implemented on the robot is relatively simple and labor-efficient. For these reasons, the data in our experiments were collected with a basic controller in the following way.

Initially, we fix the location of the hole and randomly sample the initial position of the peg from a square area of size $40mm \times 40mm$ around the center of the hole, using a constant height of $20mm$ above the rim of the hole. After that, similar to the controllers studied in [3] and [20], the automated controller executes three phases: approach, alignment, and insertion. The "approach" step is the downward movement until the tip of the peg touches the surface of the platform. The "alignment" is the step of sliding the peg on the surface of the platform along a straight line towards the center of the hole. The "insertion" step is the movement straight down; it starts after the peg is aligned with the hole. By applying the automated controller, we collected $1,000$ complete insertions composed of the 3 steps and in total tens of thousands of data points. Among these insertions, 60% are used for training, 20% for validation, and 20% for testing. Total data collection time was around 4.5 hours.

C. Results

In this section, we report evaluation results on the performance of the multi-step-ahead loss function for behavioral cloning proposed in Sec.III-B, as well as the performance of the policy with multi-modal perceptual representations described in Sec. III-A. The 7 controllers described in Sec. III-A1, III-A2, and III-A3 are trained using the multi-step-ahead loss function with horizon $K = \{1, 2, 3\}$, for a total of 21 networks. Each controller has been tested for 20 insertions on the real system, randomizing the initial conditions of both the peg position and the hole location, for a total of 420 experiments. Notice, that the hole location is not fixed anymore

and varies in the work space of the robot. The controllers that can handle different hole locations are called *goal independent*. The controllers mentioned in Sec. III-A4 were also tested, but are outside the definition of first-person view, and thus cannot generalize to different hole base locations. Therefore, for space reasons, their results are not reported. Still, it is important to mention the performance of the "EEf" controller, because this network is used in the loss function (2). There was high adherence to the data, with low values of the loss function in both the training and testing data. However, the policy performed poorly in the real experiments, because of lack of information about force and torque; the force measurements were exceeding the limits allowed by the robot, forcing the experiments to end. Nevertheless, the prediction accuracy of this controller justifies the usage of this controller in defining the multi-step-ahead loss function.

Table I reports the resulting success and failure percentages for all insertions. The table is organized in three blocks separated by solid lines, where each block is named according to the time horizon of the loss function: network IDs 1.1-1.7 are for the 1-step-ahead models, denoted as block 1; network IDs 2.1-2.7 are for the 2-step-ahead models, denoted as block 2; and network IDs 3.1-3.6 are for the 3-step-ahead models, denoted as block 3. Each block is divided into 3 sub-blocks separated by dashed lines: the first sub-block is the "ResNet" controller defined in Sec. III-A1, in the second sub-block there are the "ResNet + raw sensory data" controllers defined in Sec. III-A2, and in the third sub-block, there are the "ResNet + features extraction" controllers defined in Sec. III-A3.

1) Effect of using different sensor modalities: Table I shows that for the standard one-step-ahead loss function, in block 1, adding the end-effector and/or force/torque readings improves the performance w.r.t. using only vision, "ResNet" network ID 1.1. However, we can see that considering the end-effector information is not always useful, and the controllers that consider "ee" measurements, network IDs 1.2,1.4,1.5,1.6, actually perform worse than the controllers that are based only on vision and force/torque,"ResNet + FT raw/features". This is likely due to the mismatch of the end-effector positions between training data and the test insertions, known as distribution shift. The high information content coming from the end-effector positions comes at the price of lack of generality of the controller, which becomes tied to specific spatial

Network ID	Model Description	Full Insertion %	Partial Insertion %	Touch the Platform %	Goal Independent
1.1	1-step-ahead ResNet	30	0	70	**yes**
1.2	1-step-ahead ResNet + ee + FT	55	15	30	no
1.3	1-step-ahead ResNet + ee	50	0	50	no
1.4	1-step-ahead ResNet + FT	75	0	25	**yes**
1.5	1-step-ahead ResNet + ee + FT features	55	10	35	no
1.6	1-step-ahead ResNet + ee features	45	20	35	no
1.7	1-step-ahead ResNet + FT features	70	0	30	**yes**
2.1	2-step-ahead ResNet	95	0	5	**yes**
2.2	2-step-ahead ResNet + ee + FT	100	0	0	no
2.3	2-step-ahead ResNet + ee	75	5	20	no
2.4	**2-step-ahead ResNet + FT**	**100**	**0**	**0**	**yes**
2.5	2-step-ahead ResNet + ee + FT features	85	5	10	no
2.6	2-step-ahead ResNet + ee features	80	0	20	no
2.7	2-step-ahead ResNet + FT features	95	0	5	**yes**
3.1	3-step-ahead ResNet	85	0	15	**yes**
3.2	3-step-ahead ResNet + ee + FT	100	0	0	no
3.3	3-step-ahead ResNet + ee	100	0	0	no
3.4	3-step-ahead ResNet + FT	80	0	20	**yes**
3.5	3-step-ahead ResNet + ee + FT features	85	5	10	no
3.6	3-step-ahead ResNet + ee features	90	0	10	no
3.7	**3-step-ahead ResNet + FT features**	**100**	**0**	**0**	**yes**

TABLE I: Results of multi-modal perception, where each row corresponds to 20 insertions with random initial conditions of the peg and the hole positions. The first column is the network ID; the second column is the network model; Third to fifth column are the percentage of successful insertion; the last column indicates if the neural network model is goal independent. Full Insertion means the peg can be successfully inserted into the hole; Partial Insertion means the peg aligns with the hole, but the insertion fails; Touch the Platform represents the peg can touch the platform, but fails to align with the hole.

locations. Another advantage of not using the end-effector locations is that the controllers are *goal independent*, there is no need to specify the x, y, z location of the target hole. Indeed, for controllers not based on the "ee" measurements, we were moving the target hole freely into the whole robot workspace at test time, and performing the insertions starting from a neighborhood ($40mm \times 40mm \times 20m$) around the hole center.

2) Effect of the multi-step-ahead loss function: The idea behind the multi-step-ahead loss function is based on the findings described above. We want to exploit the high accuracy of the end-effector information during training, but avoid the high sensitivity and locality of this information at test time, which might worsen performance and limit the ability to generalize to different goal locations. By comparing all the networks in the three blocks, we find that 2-step-ahead models perform much better than the models trained by the original 1-step-ahead loss function in all the cases. Moreover, in general, the 3-step-ahead models work better than the models trained by the 2-step-ahead loss, except for network IDs 2.4 and 3.4. The "ResNet" controllers (network IDs 1.1, 2.1 and 3.1) show that vision inputs alone are insufficient to solve the task with standard 1-step-ahead loss function, having only 0.30% success rate, but using the proposed multi-step-ahead function greatly improves performance. An example of insertion trajectories for the "ResNet" controller with loss function horizons $K = \{1, 2, 3\}$ is shown in Fig.7.

3) Effect of using raw sensory data and features: In each of the three blocks, the "ResNet + raw sensory data" controllers tend to outperform the "ResNet + features" controllers independently of which sensory modalities were considered. However, the "ResNet + features" controllers have a mono-
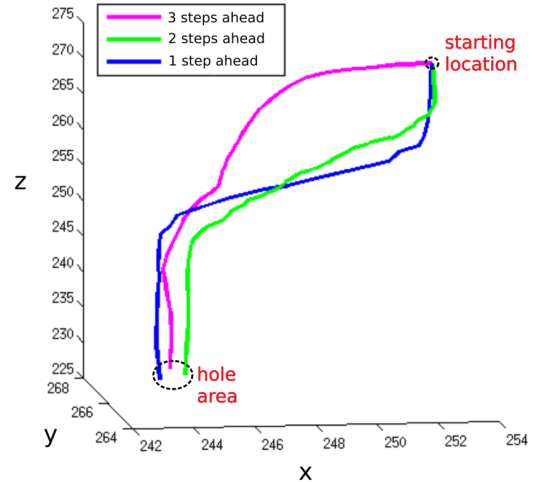

Fig. 7: Trajectories of different controllers. The blue, green and magenta lines are the 1-, 2-, and 3-step-ahead models, respectively. As indicated in the figure, the starting positions are at the top right corner, and the ending positions are at the bottom left.

tonic increase in performance as the horizon of the multi-step-ahead loss function increases (compare network ID 1.5 with 2.5 and 3.5; ID 1.6 with 2.6 and 3.6; ID 1.7 with 2.7 and 3.7). This suggests that the lower performance of these controllers in blocks 1 and 2 might be related to the need for carefully training the extra layers in the networks, which is better obtained with the longest horizon of the multi-step-ahead loss function.

According to these results, the best controllers are the "3-

step-ahead ResNet + FT features" and "2-step-ahead ResNet + FT features", achieving 100% successful insertions in the whole workspace of the robot — they both are goal-independent controllers.

V. Conclusion and Future Work

We have performed and analyzed ablation studies on multi-modal perception for peg insertion tasks on a real robotic system using a behavioral cloning approach. Some of the main conclusions we have drawn include: it is not always beneficial to combine vision, force/torque readings, and end-effector measurementd into one controller, due to the mismatch of the end-effector positions between training and testing the best performance are obtained without end-effector position and this grant goal independent controllers; it can be beneficial to map the raw force/torque and end-effector measurements to a well trained features space before using them for control; and, the proposed training method for behavioral cloning based on a multi-step-ahead loss function significantly improves the success rate of behavioral cloning.

This paper shows our initial investigations related to PiH insertion tasks validated over 400 insertions at test time on a robotic system. However, further research could be done to validate and extend our findings on multi-modal perception on real robotic systems, and further generalize the proposed multi-step-ahead loss function. Possible future work includes, but is not limited to, the following directions:

- Extend the ablation studies for multi-perception to several other assembly tasks and investigate the generalization of the proposed multi-step-ahead loss function to other learning frameworks.
- Add other sensor modalities: depth information is also a fundamental sensor input and widely used in assembly tasks [19]; the vision input from a third person p.o.v.; touch sensors on the gripper fingers.
- The method could be improved by incorporating RL, possibly using the already proposed controllers as a warm-up policy and exploring the state space to improve performance.
- It is still unclear how sensitive the drawn conclusions are w.r.t. variations in the architectures of the networks considered. In the future, we plan to add variations in our comparison framework, e.g. using different image processing network models, changing the number of fully connected layers, using different architectures, etc.

References

[1] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013. PMID: 23148415.

[2] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, May 2009.

[3] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8943–8950, IEEE, 2019.

[4] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," *CoRR*, vol. abs/1906.05841, 2019.

[5] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz, "A practical approach to insertion with variable socket position using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 754–760, IEEE, 2019.

[6] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Learning from demonstrations for real world reinforcement learning," *CoRR*, vol. abs/1704.03732, 2017.

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[8] A. Giusti, J. Guzzi, D. C. Cirean, F. He, J. P. Rodrguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, pp. 661–667, July 2016.

[9] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, pp. 343–355, 2012.

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[11] S. Reddy, A. D. Dragan, and S. Levine, "SQIL: imitation learning via regularized behavioral cloning," *CoRR*, vol. abs/1905.11108, 2019.

[12] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 4950–4957, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[13] A. Doerr, C. Daniel, D. Nguyen-Tuong, A. Marco, S. Schaal, T. Marc, and S. Trimpe, "Optimizing long-term predictions for model-based policy search," in *Conference on Robot Learning*, pp. 227–238, 2017.

[14] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models.," in *AAAI*, pp. 3024–3030, 2015.

[15] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *2018*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2062–2069, IEEE, 2018.

[16] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," *arXiv preprint arXiv:1906.05841*, 2019.

[17] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," *CoRR*, vol. abs/1501.05611, 2015.

[18] T. Inoue, G. D. Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," *CoRR*, vol. abs/1708.04033, 2017.

[19] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality tele-operation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[20] D. Ehlers, M. Suomalainen, J. Lundell, and V. Kyrki, "Imitating human search strategies for assembly," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7821–7827, IEEE, 2019.

[21] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[22] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[23] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, pp. 171–203, Jul 2011.

[24] D. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations." In Proceedings of IEEE International Conference on Robotics and Automation, 2011.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[26] M. Laskey, S. Staszak, W. Y. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 462–469, May 2016.