Abstract—Complicated assembly processes can be described as a sequence of two main activities: grasping and insertion. While general grasping solutions are common in industry, insertion is still only applicable to small subsets of problems, mainly involving simple shapes in fixed locations and in which the variations are not taken into consideration. Recently, RL approaches with prior knowledge (e.g., LfD or residual policy) have been adopted. However, these approaches might be problematic in contact-rich tasks since interaction might endanger the robot and its equipment. In this paper, we tackled this challenge by formulating the problem as a regression problem. By combining visual and force inputs, we demonstrate that our method can scale to 16 different insertion tasks in less than 10 minutes. The resulting policies are robust to changes in the socket position, orientation or peg color, as well as to small differences in peg shape. Finally, we demonstrate an end-to-end solution for 2 complex assembly tasks with multi-insertion objectives when the assembly board is randomly placed on a table.

I. INTRODUCTION

Today’s assembly tasks still rely heavily on predefined actions or movements and are limited in their ability to adapt to even small changes. This is due to the great variability in the types of problems at hand that differ in many features. Furthermore, state uncertainties in grasp alignment and hole location may result in hazardous forces due to collision with the hole surface. Using grasping and insertion together enables us to achieve most assembly tasks.

Insertion, despite the large body of work on it, is still only applicable to small subsets of tasks. Most solutions address round objects in a fixed location and do not take industrial variations into consideration. This makes current algorithms, such as the popular spiral search, unable to quickly adapt to new tasks, and they also require specific engineering for each use-case, and are costly.

Recently, researchers have tried to tackle this problem by combining reinforcement learning (RL) algorithms, known for their versatility and generality, with prior knowledge on the system, for example, learning from demonstration (LfD) or learning only a residual policy. These methods involve learning algorithms such as off policy RL [1], [2] and model-based RL [3], which are performed directly on the robot. While achieving interesting results in complex insertion tasks and demonstrating generalization properties to some extent [4], this type of online interaction is impractical on contact-rich tasks, either because data collection on real robots is extremely expensive or because unpredicted moves created by initial policies can endanger the robot and its equipment (e.g., force torque sensor), since hazardous contact forces could arise during the trials. Another RL approach is to combine a high-quality simulation with some sim-to-real algorithm [5].
While this approach can achieve generalization over different insertion tasks, its scalability is bounded by the ability to model the world via simulation. In this paper, we formulate the problem differently and solve it as a regression problem, which is a lighter methodology than RL. In addition, by combining visual input with force inputs in the decision-making process, and by adding data augmentation, we can easily handle 16 different types of pegs and plugs over different locations. Furthermore, we show that our method can handle variability in the environment such as colors or small differences in shape. Our primary contributions are:

1) A simple way to formulate an insertion problem in a regression form.
2) Backward learning, a novel approach to collect data in insertion tasks.
3) Incorporating data augmentation methods on force and visual inputs.
4) A rigorous evaluation of the method’s performance on (a) 16 different insertion tasks with spatial invariance and (b) real-life assembly tasks with consecutive insertion tasks.

II. RELATED WORK

A. Contact-rich Manipulation

Insertion problems have been extensively investigated both by the control community [6] and the machine-learning community [7]. Insertion may involve a contact model-based control or a contact model-free learning [8]. Contact model-based strategies build a contact model through analytical tools (static or dynamic), as well as statistical (GMM, HMM, SVM; [9]) approaches, followed by the delicate design of a corrective strategy, to minimize the contact force. These methods guarantee safety and efficiency and are suitable for special assembly scenarios, but are limited by the ability to model the contact.

To overcome this limitation, contact model-free strategies have been adopted, where either a Learning from Demonstration (LfD; [10]) or Learning from the Environment (LfE; [11]) approach is used. The LfD approaches benefit from expert demonstrations and enable effective, rapid skill acquisition in physical robots. The recently proposed method for imitating human compliant behaviors using different mapping techniques and without pre-programming is a prime example of LfD’s utility [12]. Similarly to contact model-based approaches, LfD guarantees safety and efficiency, but its flexibility is limited by the expert demonstrations.

Model-free RL has proved useful for solving challenging tasks in a wide range of problems [13]. However, manipulation tasks are especially challenging due to their continuous action space and their high-dimensional state space [14], resulting in large amounts of data collection and long training times [14]. Although recent progress in RL has led to significantly better sample efficiency, algorithms without additional prior knowledge tend to have safety and robot wear-and-tear issues.

In order to overcome these issues, several approaches tried to exploit prior knowledge on the tasks. For example, residual policy methods use the knowledge on the task to pre-program a fixed policy and use RL only to learn a correction policy. This method was compared to pure RL on a connector insertion task showing improved performance and reduction in learning time [15]. Another alternative is using LfD to derive a baseline policy [16] or to initiate the learning process [15], [17], [18], thereby avoiding some of the safety risk of initial random policy. The authors of [16], which used LfD to learn a baseline policy and RL to learn a residual policy correction, showed interesting results in peg, gear and LAN cable insertion. While the above-noted approaches have shown significant results in very challenging problems, online RL algorithms in robotics tend to be impractical in contact-rich tasks.

An alternative approach for enabling deep-RL on robots is Sim2Real: learning in a simulation and transferring the learned policy to a real robot. Such an approach has shown promising results, most notably, in grasping [19] and, most recently, insertion tasks [5]. Sim2Real highlights the generalization ability of RL approaches in insertion tasks if a large amount of diverse data can be collected. However, this approach is limited by the domain expert’s ability to model the real-world scenario in the simulation.

B. Multimodal Representation Learning

Force and visual information are two complementary modalities. While force feedback is statically more informative as the overlap between the peg and hole grows [5], the visual information is diluted as the occlusion increases. On the one hand, high-precision assembly tasks [17] must use force information for the fine-tuning of the manipulation insertion. On the other hand, force information can be useless when there is no overlap between the peg and the hole. This is common in electric plugging (e.g., AC or Aux), where the socket holes are smaller than the system error in localization, control, or grasping. Recent works on insertion problems demonstrated the multi-modal approach’s ability to solve flexible and rigid insertion tasks [1] with a certain degree of generalization [4]. However, both [1] and [4] used fixed cameras to add visual information to the system, thereby increasing the space complexity and decreasing the algorithm’s ability to generalize over space. In addition, fixed cameras may insert noise to the system in the form of occlusions. A more suitable approach for insertion is to use a wrist camera at 45 degrees (relative to the End Effector) directed at a point between the two fingers.

III. METHOD

The proposed approach is composed of two steps. The first step entails localizing the holes, sockets, threads, etc. (herein “objectives”) in the scene and using a PD controller to follow the calculated path. We refer to this step as ”Localization Process”. In the second step, a residual policy, denoted by \( \pi_{\text{residual}} \), is incorporated on top of the main policy. In contrast to other residual policy methodologies, where the policy is added at any state (e.g., [15], [20]), which results in the learning of large actions far away from the hole opening [2], our residual policy is activated only when there is a large gap between the planned kinematics and the actual robot kinematics. In an insertion task, the kinematic gap is due to the collision with the hole surroundings creating a high jerk (or force change).
Unlike other methodologies that use only force information in the residual policy (e.g., [5], [16]) or only vision information (e.g., [13]), we use a multi-modal approach, fusing together force and vision information. In this work, we do not use RL methods to learn the residual policy but rather formulate the problem as a regression problem. This provides us with an offline learning advantage without the risk of damaging the robot or its surroundings. To collect the data efficiently, we use the backward learning approach described in Section III-C.

A. The System’s Components

We use a Franka Emika Panda arm (Figure 4) with 3 sensory inputs. The former is a RealSense D435 wrist camera, horizontal w.r.t. to the End Effector (EEF for short); the latter, another D435 camera, tilted in 45° and focused at a point between the EFF fingers. The third input is force sensing on the EEF, where we use a force estimator calculated by the robot.

The horizontal camera is needed to localize the objectives, while the 45° camera provides fine visual information close to the interaction area. Fused with the force sensing, this combined information stream is conveyed to the residual policy. A scheme of this architecture is presented in Figure 6.

B. Formulating the Residual Policy in a Regression Form

The state of the trained agent is composed of two components. The first component is an image of the size $H \times W \times C$, which correspond to the height, width, and number of channels, respectively. The second component is a vector that describes the force and moment, denoted by $F \triangleq (f_x, f_y, f_z) \in \mathbb{R}^3$ and $M = (m_x, m_y, m_z) \in \mathbb{R}^3$, respectively.

To avoid occlusion and to improve our ability to generalize to different locations, the images are taken from the 45° camera (from the robot wrist). We note that using the camera in such a way offers a crucial advantage, since generalization is easier from the EEF ego perspective than from the fixed cameras. In addition, the input state space is expressed in the EEF coordinate system.

The action (i.e., output of the NN) is the real robot’s movement in Cartesian space (since it is advantageous in contact-rich tasks [21]). Unlike other methods, we express the action as a one-step correction movement that results in success, thereby decoupling the control rate and the action. We denote by $\Delta x, \Delta y$ and $\Delta z$ the desired correction needed for the EEF in the Cartesian space w.r.t. the current location. In addition, $\Delta \theta_x, \Delta \theta_y$ and $\Delta \theta_z$ are the spherical orientation corrections (executed in Quaternion coordinates). We define $\Delta = (\Delta x, \Delta y, \Delta z, \Delta \theta_x, \Delta \theta_y, \Delta \theta_z)$. To summarize, in our regression problem, the algorithm learns the desired $\Delta$ based on the input. The objective is to find a policy that maps a tuple of (image, force sensing) → action (i.e., $\Delta$). Formally, we look for $\pi_{\text{residual}} : \mathbb{R}^{H \times W \times C} \times \mathbb{R}^6 \rightarrow \mathbb{R}^6$ with an architecture of the type presented in Figure 6. For regression, we need to define the desired output. Specifically, we define with $D \triangleq (\Delta x^d, \Delta y^d, \Delta z^d, \Delta \theta_x^d, \Delta \theta_y^d, \Delta \theta_z^d)$ the labelled (or desired) action to be applied. Therefore, the loss is defined by

$$\text{loss} \triangleq \| \Delta - D \|^2. \quad (1)$$

Below we describe how to collect $D$ (Section III-C).

In Figure 5, we demonstrate the importance of choosing the EEF coordinate system in combination with the 45° camera. First, if one opts for a fixed camera, the two different positions presented would result in two different states, with the only difference being in the POV. Second, if one chooses the robot coordinate system instead of the EEF, the action in the two cases would be different.

C. Backward Learning for Data Collection

In order to collect the data for the insertion regression problem, we propose to do a Backward Learning, where we initiate our data collection process by putting the robotic arm in the final state; namely, in a state where the peg or plug is positioned inside the hole. We record this position and define the close proximity to the goal state $L_{\text{pos}}$. Now that we have the final state recorded, our goal is to generate collisions around the hole in order to learn $\Delta$.

We define $\bar{r} \triangleq (x, y, z)$ and $\bar{\theta} \triangleq (\theta_x, \theta_y, \theta_z)$. Let us define the close proximity to the goal state $L_{\text{pos}}$ as a disk

Fig. 4: The system’s components include 2 RealSense D435 wrist cameras: (1) horizontal w.r.t. to the EEF. (2) tilted in 45° and focused at a point between the EFF fingers.

Fig. 5: Different angles and locations of a socket board with the same $\Delta Y$ in the EEF coordinate system. In the corresponding ellipses, the tilted 45° camera is similar. Note that the state is different when the image is captured from a fixed camera, but from the tilted 45° camera the captured image is almost identical. In addition, the action when represented in the EEF coordinate system is similar.
motion (i.e., surface. We determine whether a collision occurs during the position constant. In addition, when analyzing a simple scheme, influences the wrist camera’s position and distance from the camera.

Next, we uniformly sample target points from $B_{xy} \times B_\theta$ and denote them with $\{p_i\}_{i=0}^{N_p}$. We define $F_{th}$ as a force threshold and $M_{th}$ as a moment threshold, both measured on the EEF. For each $p_i$, first the robot reaches an area above the hole $p_i + z_{max}$ then it tries to reach $p_i$ while measuring the force $F$ and moment $M$ along the trajectory. We note that the arm does not necessarily reach $p_i$, due to collisions with the hole surface. We determine whether a collision occurs during the maneuver (i.e., $F \geq F_{th}$ or $M \geq M_{th}$) and if so, calculate the state and the action. Specifically, for the state, we measure the registered $45^\circ$ camera image, and (2) the force and moment vector. For the action, we calculate a corresponding $\{D_i\}_{i=0}^{N_p}$, where $D_i \triangleq \{x^i, y^i, z^i, \theta_x^i, \theta_y^i, \theta_z^i\} - L$, i.e., each $D_i$ is a corrective term that needs to be applied in order to bring the EEF to the correct position and pose. To decrease the wear and tear on the robot we choose low values for $F_{th}$ or $M_{th}$ which result in delicate collision with the hole surroundings. Summarizing, we have input samples $\{p_i\}_{i=0}^{N_p}$ that we wish to regress to output corrective actions $\{D_i\}_{i=0}^{N_p}$. This procedure is presented in Algorithm 1.

The motivation for the proposed Algorithm 1 and Procedure 2 is as follows. In insertion tasks, the common errors that result in collision are due to grasping misalignment and errors in visual localization. A grasping misalignment can create a delta error in $\theta_x$ and $\theta_y$ in the EEF coordinate system between the hole’s plane and the peg. Errors in the localization mainly stem from the visual localization, specifically, in the $x$, $y$, and $\theta_z$.

### D. Data augmentation

In order for this algorithm to be applicable to an industrial environment, it should have generalization capabilities as well as robustness to small variations in the environment, e.g., colors, shapes, illuminations, etc. To overcome these challenges, we introduce various augmentation processes to the visual input as well as the force input.

**Visual Augmentation:** We divide visual augmentations into color and shape augmentations. To overcome variation in peg color or variance in the illumination, we combined three different visual augmentation algorithms. The first is random convolution, which was first introduced in [22], augments the image color by passing the input observation through a random convolutional layer. The second is color jitter, which converts the RGB image to a HSV colormap and adds noise to the HSV channels, resulting in perturbations of brightness, contrast, and color. The third augmentation algorithm, gray scale, converts RGB images to grayscale.

In order to overcome variation in the peg shape, we propose two shape augmentations. The first is translation, where a random translation renders the full image within a larger frame, resulting in the original image appearing in a random position across the larger frame. For example, a $100\times100$ pixel image could be randomly translated within a $108\times108$ empty frame. The second is cropping, in which a random rectangle extracts a random patch from the original frame.

All these augmentations are used simultaneously, as presented in Figure 7. We note that since the peg’s shape influences the wrist camera’s position and distance from the holes in the collision, cropping and translation is required to compensate for these errors.

**Force and Moment Augmentation:** The force and moment measurements are strongly dependent on the collision force, which is a product of the main policy velocity, and the transition constant. In addition, when analyzing a simple scheme,

---

**Algorithm 1:** Backward Learning for Data Collection

**Input:** Maximum Height $z_{max} = 50[mm]$; Force threshold $F_{th}$; Momentum threshold $M_{th}$

1. **Init:** Insert peg/plug into the hole, save final pose $L$
2. **Set:** $(\tilde{r}_0, \tilde{\theta}_0) = L$, States $= \emptyset$, Actions $= \emptyset$
3. for $i \leftarrow 1, \ldots, N_p$, do:
   4. $(\tilde{r}^i, \tilde{\theta}^i) = RDG(\tilde{r}_0, \tilde{\theta}_0)$ # Procedure 2
   5. $T = (\tilde{r}^i, \tilde{r}_0^i, \tilde{r}_0^\theta, \tilde{\theta}_0^x, \tilde{\theta}_0^y, \tilde{\theta}_0^z)$ # $\tilde{r}_0^i$ located in hole
   6. Move robot to state $T + z_{max} \#$ to start point
   7. Plan trajectory and move robot to state $T$
   8. while Robot did not reach $T$ do:
      9. Measure force $F$ and moment $M$
      10. if $F \geq F_{th}$ or $M \geq M_{th}$ then:
         11. Measure $(x, y)$ and $(\theta^x, \theta^y, \theta^z)$
         12. $D_i = \{x, y, \theta^x, \theta^y, \theta^z\}$
         13. States $\leftarrow \{45^\circ$ camera, $F, M\}$
         14. Actions $\leftarrow \{L - D_i\}$
      15. break while loop
   16. end
   17. end
18. return States, Actions

**Procedure 2:** Random Data Generator (RDG)

**Input:** Maximum Randomized Location $b_0 = 10[mm]$; Maximum Randomized Angle $\theta_0 = 10^\circ$; Initial $\tilde{r}_0$; Initial orientation $\tilde{\theta}_0$

1. $\tilde{r} \sim Uniform[\tilde{r}_0 \pm b_0]$
2. $\tilde{\theta} \sim Uniform[\tilde{\theta}_0 \pm b_0]$
3. return $\tilde{r}, \tilde{\theta}$
one arrives at the conclusion that the direction of vectors \( (F, M) \), not their magnitude, is the most valuable factor in insertion tasks. Therefore, we incorporate force augmentations, achieved by simply multiplying both the force and the moment by a random constant \( \alpha \): 
\[
F_{\text{augment}} = \alpha F, \quad M_{\text{augment}} = \alpha M \quad \alpha \sim U[0, 1].
\]
To the best of our knowledge, we are the first to do so in such a setup.

E. Solving the Angular Camera Ambiguity Problem

Recovering the distance between two points in world coordinates without depth information using a single image is an ill-posed problem that results in ambiguity \[23\]. Recently, convolutional networks demonstrated depth prediction from a single image by learning priors about objects and their shapes \[24\], \[25\]. However, single image methods are limited in their ability to generalize to previously unseen types of images. This ambiguity can be solved if the depth from the camera is known. But realizing depth sensing in insertion tasks is rather challenging, because most sensors lack the resolution needed to distinguish between two objects from a relatively close distance. Moreover, even if such sensors can be found, they do not resolve the occlusions issues during the task. We propose to capture each input image only when the plug touches the hole surface. In this scenario, the hole opening and the tip of the peg are aligned on a single plane and the network can easily learn the distance using only a single RGB image.

F. Rough Localization Using Iterative ORB Algorithm

Localization can be a computationally intensive task. However, industrial use cases are often limited in their variability and predictability. Moreover, in assembly tasks, for example, products are often composed of a few components connected to a rigid board. We use the ORB algorithm \[20\] due to its simplicity and agility. In our extensive experiments, we found that in most cases there is no need for any more-advanced localization algorithms.

This rigidity assumption reduces the complexity of the localization problem to the task of finding the board’s center, location and orientation. Afterward, the locations of the sockets and electric components relative to the center are easily retrieved. We used the ORB algorithm to compare the current image with an image taken a-priory. Before taking this image, we localized each important component in the environment, calculating the delta between the components and the EEF’s location (when the pre-taken image was captured). To avoid camera calibration, this process is applied iteratively in a closed loop until the two images match. When the two images match, the delta between the EEF location and the holes should match (with some localization error) to the pre-calculated delta. Thus, we can retrieve the holes location by summing the current EEF position and the pre-calculated delta.

IV. EXPERIMENTS

In this section, we describe the experiments conducted to validate our methodology, in which we address the following questions:

1) Scalability - How does our algorithm scale to various insertion problems? How many samples are needed? What is the average time point when the object should be inserted?
2) Spatial Invariance - Can we obtain similar results in other locations?
3) Generalization - Can we generalize to similar types of plugs with different colors and shapes?
4) Transferability - How well does the trained policy transfer to other tasks?

A. Experimental setup

Data Collection and Training: In the first stage, we collect 100 data points (5 minutes for each plug) using the backward learning approach (Algorithm 1). In the second stage, we train a residual policy neural net architecture for each plug (described in Figure 6) with the proposed visual and force augmentations of Section III-D. Based on 100 samples, we generate 640,000 training samples (10,000 repeats on 64 samples batches) using 2 GPU GeForce RTX 2080 Ti. The total training time takes 40 minutes.

Testing Procedure: We test our method in the system described in Section III-A following a similar procedure to the one described for Algorithm 1. In each experiment, we examine one plug-socket setup. We generate test points,
denoted by $N_{\text{test}}$, around the testing socket in a similar manner to lines 5-9 in Algorithm 1 by uniformly sampling $|N_{\text{test}}|$ target point $T$ (line 9) from $B_{zxy} \times B_{\theta}$. We note that $T$ is the localization point and that it may contain errors from grasping, misalignment and localization. Afterward, the robot sends $T$ to the process described in Algorithm 3. The PD controller executes each movement command for $0.4\text{[s]}$. A trial is considered successful if the robot inserts the peg/plug into its designated hole with tolerance $z - z_0 < 1\text{[mm]}$. An unsuccessful trial is when the duration it takes to complete the task exceeds $10\text{[sec]}$. We repeat each insertion trial 200 times (with random initialization) in order to achieve statistically significant results. In order to examine realistic scenarios, the pegs are not fixed to the gripper (allowing slippage between the peg and the fingers) during insertion, and the 200 trials are consecutively applied in a row to intentionally create misalignments during the insertions.

B. Scalability for different insertion setups

In order to evaluate our method’s ability to scale to 16 different tasks, we collected 16 different real-life insertion and threading tasks with different properties. The tasks were split into two groups: insertion tasks and threading tasks. In insertion tasks, the robot holds the peg and tries to insert it into a designated hole. In threading tasks, the robot holds a shaped hoop in its gripper and threads it over a shaft. The insertion tasks include 10 different pegs/plugs (AC 2-pin, AC 3-pin, Aux, DC12V, Ethernet RJ45, HDMI, Keys, Power cord, USB, and USB-C) with different properties that affect the insertion process; for example, different shapes (e.g., non-convex shapes, like an electric plug, vs. convex shapes, like a USB-c plug), clearance sizes (e.g., 0.1mm in HDMI vs. 1mm in AUX) and sizes (2.5mm in Aux vs. 20mm in power chord). The threading tasks include 6 different shapes: 5 wooden shafts (triangular, cubical, cylindrical, rectangular, and half circle) and one plug (VGA). In addition, the threading problems varied in many properties, such as friction (wood vs. steel) shape (triangular, cubical, etc.) or clearance (0.1mm for the cubical shapes, 1mm for the triangular, etc.).

Figure 8 summarizes the success rate and time for completion of the 16 different insertion cases, averaged over 200 trials. The insertion-oriented tasks (marked gray) scored 95.9\%, on average, over all the tasks. The lowest score in the insertion tasks was 90.5\%, in the keys and HDMI tasks, while the highest score was 100\%, in the AC 2-pin, AC 3-pin and Aux. The insertion-oriented tasks’ average trial duration was 2.9[sec]. The longest time duration was 5[sec], with the HDMI plug, and the shortest was 1.3[sec], with the AC2 pin. The threading-oriented tasks (marked orange) scored an average of 96\% over all the tasks. Here, the lowest score was 78.5\%, in the VGA task, and the highest was 100\%, in the circle, half circle, triangle, and cubical wooden threads. The threading-oriented tasks’ average duration was 3.0[sec] over all the tasks. The longest duration was 4.7[sec], with the VGA plug, and the shortest was 2.2[sec], with the cubical thread.

During the experiments, we noticed three reasons for the variance in the success rate and time. First, the methods encountered difficulties in small clearance, necessitating force (such as HDMI and a power cord). In those cases, small errors might result in jamming. Second, another challenging type of scenario was when the hole and clearance were small, such as in USB-C and DC12V. Those cases require very fine control, as the robot can move over the hole without inserting the peg. Third, the method had a difficulty tackling an insertion where the hole and the plug are of similar color, such as a HDMI and power cord. In these cases, visually separating between the plug and socket is hard. The VGA task was, by far, the task with the lowest scores. During the experiments, we noticed that the combination of small clearance in the VGA with the high friction between the peg and the hole surface results in many instances of the peg becoming jammed or the controller getting stuck due to friction.

C. Sample efficiency

In this section we assess how many samples are required to successfully accomplish an insertion task. In order to calculate the relation between the number of samples and the success rate or time for completion, we trained 8 different policies using 10, 15, 20, 25, 30, 50, 100 and 200 samples. Each policy was tested 50 times on the AC 2-pin plug task. The success rate and time for completion vs. samples required is depicted in Figure 9. The success rate rapidly increases between 0 and 50 samples, reaching 100\% in 50 data points, while the time for completion steeply declines between 0 and 100 samples, reaching a minimum of 1.3[sec] at 100 samples. Based on this evidence, we can re-verify that 100 samples are sufficient for the Section IV-B experiments.

D. Generalization

Generalize Over Location and Orientation: To evaluate our method’s ability to generalize over space, we preformed the AC 2-pin plugging in four different locations and orientations, by placing the socket board on the four corners of the robot table. Each experiment was repeated 50 times and the success rate and time for completion were measured. The results indicate no decline in the success rate, but the experiment took 30\% longer, on average, than in the learned location experiment.

Generalize Over Shape: To evaluate our method’s ability to generalize over plug shapes, we trained a policy with an AC 2-pin plug and tested the performance on different AC 2-pin plug shapes (figure 9). Our results indicate a minor decline (1 unsuccessful experiment) in the success rate and an 60\% increase, on average, in the duration time in comparison to the basic plug. We note that without visual augmentation our model success rate drops to 41\%.

Generalize Over Color: To evaluate our method’s generalization over colors, we trained a policy with a black AC 2-pin plug and tested the performance on a white AC 2-pin plug (figure 9). The results indicate a minor decline (two unsuccessful experiments) in the success rate and a 100\% increase, on average, in the duration time where without visual augmentation our model success rate drops to 18\%.

Transferability: In this section, we evaluated how well a
trained policy is transferred to other tasks. The policy was trained over 100 samples on the AC 2-pin task. We evaluated how many additional samples from the new task are needed to accomplish the task. The new task was chosen so as to be similar to the trained task in order to be able to examine whether the algorithm captures the commonality between the tasks. We chose as the new task a 2-pin plug as well, but it required a different grasping method and had a significantly different shape (figure 9 right plug). We trained 5 different policies: adding 0, 5, 10, 15 and 20 samples. Trivially, the algorithm without any additional sample, as expected, score 0%. Notably, only five samples were required to achieve a score of 70%, and 10 for a score of 100%, in comparison to 50 without the pre-training policy. This implies that roughly only 30[sec] of data collection is needed to learn new insertion tasks using our method.

Fig. 9: Closely related problems. Four different electric plugs demonstrating real-life variations in shape, color, and other properties.

**V. END-TO-END ASSEMBLY TASKS**

In this section, we tested our method on complex assembly tasks, from grasping to hole localization, to insertion. The tasks goal was to test the method ability to learn a variety of insertion tasks with only one policy, to handle consecutive insertion tasks even in close proximity, and to combine both the localization process and the insertion process. In the following, we successfully demonstrate these abilities on two environment designs.

**First Environment Design** (Figure 3b up): A board with 3 AC sockets, 2 USB sockets, and an Ethernet socket is presented. The robot’s goal is to insert all the plugs into their designated holes, one after the other, with only one policy. We emphasize that after the insertion of each plug, the plug is left in the hole, which changes the environment’s state and disturbs the insertion of the other plugs. In order to evaluate our method’s robustness and generalization, the plugs were of different colors and shapes; in addition, the USB and Ethernet cord remained attached, thus creating occlusions.

**Second Environment Design** (Figure 3b bottom): A wooden threading game that includes five different wooden shafts and 20 rings with matching holes. The goal of the game is to insert the different shapes into their matched shafts. This game included the following pieces: 5 half circles, 4 rectangles, 3 circles, 2 squares and a triangle. To solve the environment, the robot needs to insert all 15 pieces into their matched holes.

Fig. 10: The success rate (blue, circles) and time for completion (orange, squares) given number of training samples. Given enough training samples we see that the system achieves the minimal time for plugging and 100% success rate.
Training Procedure: We begin with collecting 100 data points from each assembly component (e.g., 300 data points for the first environment) in order to learn a general residual policy. Afterward, we save the holes’ and shafts’ locations in advance and register an image (together with the EEF location) using the horizontal camera for the localization process (section III-F). Finally, we place the plugs on a conveyor, recording each of the plugs’ location for grasping purposes.

Testing Procedure and results: Before starting the process, we randomly place the socket board on the robot table. Then, the localization process (as describe in Section III-F) starts by roughly localizing the holes and saving each of their locations. After localization, the robot enters an operational loop, in which it reaches, grasps, and inserts a plug into its designated hole, until all the objects are inserted. In both settings, all plug-ins were successfully inserted.

VI. CONCLUSIONS

We introduced a novel framework for learning a wide range of insertion tasks. We address the wear-and-tear and safety issues in contact-rich tasks by formulating the residual policy in a regression form. In addition, we introduce a backward approach for the fast collection of data without endangering the robot. In addition, we used a multimodal approach that combines vision and force sensing but replaces the commonly used fixed camera with a 45 deg wrist camera for better generalization.

Our results show that an agent can reliably solve 16 different real-life insertion tasks and is robust to socket pose and variance in plug color or shape. Finally we demonstrate our framework’s ability to tackle two real-life assembly tasks with multi-insertion objectives. Qualitative results can be found in our supplementary video on our website: https://sites.google.com/view/insertionnet/

ACKNOWLEDGMENT

We thank the anonymous reviewers and Mrs. Dana Rip for their very helpful comments that improved this manuscript.

REFERENCES

[1] M. Vecerik, O. Sushkov, D. Barker, T. Rothöl, T. Hester, and J. Scholz, “A practical approach to insertion with variable socket position using deep reinforcement learning,” in ICRA, 2019.
[2] M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox, “Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning,” in ICRA, 2020.
[3] 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 IROS, 2018.
[4] 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 ICRA, 2019.
[5] O. Spector and M. Zacksenhouse, “Deep reinforcement learning for contact-rich skills using compliant movement primitives,” arXiv preprint arXiv:2008.13223, 2020.
[6] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, “Automatic synthesis of fine-motion strategies for robots,” The International Journal of Robotics Research, vol. 3, no. 1, pp. 3–24, 1984.
[7] V. Gullapalli, A. G. Barto, and R. A. Grupen, “Learning admittance mappings for force-guided assembly,” in ICRA, 1994.
[8] J. Xu, Z. Hou, Z. Liu, and H. Qiao, “Compare Contact Model-based Control and Contact Model-free Learning: A Survey of Robotic Peg-in-hole Assembly Strategies,” 2019, arXiv: 1904.05240.
[9] K. P. Murphy, Machine learning: aprobabilistic perspective. MIT press, 2012.
[10] S. Schaal et al., “Learning from demonstration,” Advances in neural information processing systems, pp. 1040–1046, 1997.
[11] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” IJRR, 2013.
[12] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” Robotics and Autonomous Systems, 2009.
[13] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.
[14] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al., “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” arXiv preprint arXiv:1806.10293, 2018.
[15] 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:1906.05841, 2019.
[16] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, “Residual learning from demonstration: Adapting dynamic movement primitives for contact-rich insertion tasks,” arXiv e-prints, pp. arXiv–2008, 2020.
[17] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, “Deep reinforcement learning for high precision assembly tasks,” in IROS, 2017.
[18] K. S. Luck, M. Vecerik, S. Stepputtis, H. B. Amor, and J. Scholz, “Improved exploration through latent trajectory optimization in deep deterministic policy gradient,” in IROS, 2019.
[19] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Boumanalis, “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks,” in CVPR, 2019.
[20] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” in ICRA, 2018.
[21] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, “Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks,” in IROS, 2019.
[22] K. Lee, K. Lee, J. Shin, and H. Lee, “Network randomization: A simple technique for generalization in deep reinforcement learning,” in ICLR, 2019.
[23] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” NIPS, 2014.
[24] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in ICCV, 2015.
[25] F. Liu, C. Shen, G. Lin, and J. Reid, “Learning depth from single monocular images using deep convolutional neural fields,” IEEE TPAMI, 2015.
[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in ICCV, 2011.