2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)

Improving the Generalizability of Robot Assembly Tasks Learned from Demonstration via CNN-based Segmentation

Iñigo Iturrate[1, 2], Etienne Roberge[3], Esben Hallundbæk Østergaard[2], Vincent Duchaine[3], and Thiusius Rajeeth Savarimuthu[1]

[1]Maersk McKinney Moller Institute, University of Southern Denmark Campusvej 55, 5230 Odense M, Denmark
[2]Universal Robots A/S, Energivej 25, 5260 Odense S, Denmark
[3]Control and Robotics Laboratory (CoRo), Faculty of Automated Production Engineering, École de Technologie Supérieure, Montreal, Canada

***Abstract*— Kinesthetic teaching and Dynamic Movement Primitives (DMPs) enable fast and adaptable learning of robot tasks based on a human demonstration. A task encoded as a dynamic movement primitive can be reused with a different goal position, albeit with a resulting distortion in the approach trajectory with regards to the original task. While this is sufficient for some robotic applications, the accuracy requirements for assembly tasks in an industrial context, where tolerances are tight and workpieces are small, is much higher. In such a context it is also preferable to keep the number of demonstrations and of external sensors low. Our approach relies on a single demonstration and a single force-torque sensor at the robot tool. We make use of a Convolutional Neural Network (CNN) trained on the force-torque sensor data to segment the task into several movement primitives for the different phases: pickup - approach - insertion - retraction, allowing us to achieve better positional accuracy when generalizing the task primitives to new targets. To the best of our awareness, we are the first to utilize a CNN as a segmentation tool to improve the generalization performance of DMPs.**

I. Introduction

The appearance of collaborative robots has significantly changed the field of industrial robotics by opening it up to an audience that traditionally would have lacked the technical expertise to program these robots. Co-bots are cheaper, and they act as tools that can work alongside humans.

As the number of users increases, collaborative robotics faces the challenge of further lowering its entry barrier by improving on usability and flexibility. Programming by Demonstration (PbD) has shown promise here as an intuitive and user-friendly programming method, which inexperienced users prefer over graphical programming [1]. However, its performance has not achieved the robustness needed for industrial applications [2], and apart from Gao et al. [3] it has not been extensively benchmarked in an assembly context.

Within programming by demonstration, force-controlled approaches to manual guidance of the robot during teaching are common, as they allow the user precise control over the movement of the robot end-effector with the feeling of reduced mass [2]. This method has been incorporated into industrial robots such as the Kuka LBR iiwa. A similar control is available from Robotiq for the Universal Robots arms as *ActiveDrive*, provided as part of the *Force Copilot* software [4]. However, methods like *ActiveDrive* typically only allow positional replay of the demonstration, which might be unsuitable for assembly.

On the other hand, frameworks such as Dynamic Movement Primitives (DMPs) have enabled effective learning and generalization of a variety of tasks, including force-control [5] and assembly [6]. Still, they have not been extensively tested in an industrial setting, where the extreme accuracy requirements may result in failure when trying to generalize primitives to new contexts.

In this work, we propose to use Convolutional Neural Network (CNN)-based segmentation of the insertion location to improve the accuracy of DMP generalization. Basing the implementation on DMPs allows local modifications to each of the task phases, e.g. speed-up of the approach, pickup and retraction motions, while slowing down the insertion to reduce contact forces. To verify our approach, we perform a series of experiments on four different workpieces.

II. Approach

Dynamic Movement Primitives [7], is one of the most widely-used frameworks for encoding robot trajectories learned from demonstration. Other approaches, such as statistical learning based on Gaussian Mixture Modeling and Hidden Markov Models [8] require multiple demonstrations, which may not be feasible in an industrial context, where setup time has a significant influence on cost.

The advantages of DMPs include one-shot learning, and the ability to generalize the primitive to new goal configurations [9] without the need for additional demonstrations. The compact mathematical representation can be extended with additional terms for, e.g. obstacle avoidance [10]. Hybrid position-force control can also be added [11], and the forces can be learned iteratively while progressively improving task replay speed and minimizing assembly contact forces [6].

Due to the system dynamics in DMPs, all movements will reach zero velocity at the goal configuration. However, the forcing terms of two overlapping DMPs can be modified to smoothly transition from one to the other while maintaining a continuous velocity profile [12]. By changing the underlying phase-controlling canonical system, movement primitive sequencing can be performed online [13]. This enables us to use task segmentation to achieve better generalization by learning and combining separate primitives for pick-up, insertion, and other phases of the task.

CNN architectures have been used in a wide-ranging amount of work in recent years and have proven to be a remarkable feature learning technique in numerous fields of study, including the classification of temporal signals in fields like speech recognition [14] or in video [15].

When applied to programming by demonstration, the use of CNNs has mainly revolved around video recognition, where the task is taught by showing movements and gestures through camera feeds [16]. CNNs have also been mixed with DMPs to create intelligent replay strategies by learning task specific features [17] by using images of the environment.

In this work, CNN classification is used to segment a kinesthetic demonstrations of an assembly task. By analyzing the forces applied on the robot during admittance control for a demonstration, it is possible to detect that an insertion task has been performed [18]. We propose to use this method to generate a sequence of DMPs focused on the assembly task. This way, changing the position of the insertion tasks during replay will only affect the primitives used for carrying the workpiece from the pick-up to the insertion point and the retraction from the insertion. The insertion primitive itself will remain unmodified and will simply be translated in the workspace.

Our approach relies solely on a force-torque sensor located at the robot tool. This is a common feature of collaborative robots, or can otherwise be integrated relatively inexpensively. This contrasts with other approaches within PbD, which rely heavily on computer vision or multi-sensory feedback. While such approaches have other advantages, they often require careful sensor or camera calibration, which limits their use to controlled environments. This can significantly raise the cost and complexity of the installation, making it infeasible for small and medium-sized enterprises.

III. Dynamic Movement Primitives

A. Mathematical Model

DMPs are typically expressed as two second- or third-order dynamical systems: a transformation system (equations 1 - 3) and a canonical system (equation 6). Here we use the formalization established by Kulvicius et al. [13], which utilizes a sigmoidal canonical system, and introduces a dependence on time.

1) Position Transformation System: Our transformation system is expressed as a third order system:

$$\tau \dot{z} = K(r - y) - Dz + f(v), \tag{1}$$

$$\tau \dot{y} = z, \tag{2}$$

$$\tau \dot{r} = \begin{cases} \frac{\Delta t}{T}(g - s), & \text{if} \quad t \leq T \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Equation 1 can be viewed as a spring-damper system with spring constant K and damping coefficient D. Here, $\dot{z}$, z and y represent the acceleration, velocity and position, respectively. The non-linear term $f(v)$ influences the trajectory shape. Time constant τ allows speed-up or slow-down of the system. The goal signal r is ramped according to equation 3, to avoid problems with high initial accelerations. In equation 3, g and s are the goal and start positions of the movement, Δt is the time-step, t is the current time, and T is the total execution time of the primitive.

2) Orientation Transformation System: We define a quaternion as $\boldsymbol{q} = [q_w, q_x, q_y, q_z]^\intercal$, where q_w is the real part and q_x, q_y, q_z are the imaginary parts. We define the orientation error between two quaternions as: $\boldsymbol{e_{(1,2)}} = \boldsymbol{q_2 q_1^{-1}}$, where $\boldsymbol{q}^{-1}$ is the inverse quaternion.

An angular velocity $\boldsymbol{\omega}$ is a quaternion with no real part, i.e. $q_w = 0$. From Pastor et al. [11], the orientation equivalent for equation 1 is then:

$$\tau \dot{\boldsymbol{\omega}} = -\boldsymbol{K} \cdot \left[0, e_{(r,q)_x}, e_{(r,q)_y}, e_{(r,q)_z}\right]^\intercal - \boldsymbol{D} \cdot \boldsymbol{\omega} + \boldsymbol{f}(x). \tag{4}$$

An angular velocity $\boldsymbol{\omega}$ produced by 4, and a current orientation quaternion $\boldsymbol{q}$ correspond to a quaternion derivative:

$$\dot{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{\omega} \cdot \boldsymbol{q}. \tag{5}$$

The term $\boldsymbol{r}$ in equation 4 is the quaternion equivalent of equation 3. For this, we use SLERP to linearly interpolate orientations between the goal $\boldsymbol{g}$ and start $\boldsymbol{s}$ orientations [19].

3) Canonical System: The temporal evolution of the system is controlled by the canonical system:

$$\dot{v} = -\frac{\alpha_v e^{\alpha_v (T + t_{delay} - t)}}{\left(1 + e^{\alpha_v (T + t_{delay} - t)}\right)^2}. \tag{6}$$

The constant α_v controls the steepness of the sigmoidal decay at time $T + t_{delay}$. We introduce t_{delay} to prevent any significant decay before time T has been reached.

4) Forcing Term: The non-linear term $f(v)$ is parameterized as:

$$f(v) = \frac{\sum_{i=1}^{n} \psi_i w_i v}{\sum_{i=1}^{n} \psi_i}. \tag{7}$$

This is a sum of weighted Guassian kernels ψ given by:

$$\psi_i = e^{-\frac{\left(\frac{t}{T} - c_i\right)^2}{2\sigma_i^2}}, \tag{8}$$

where c_i and σ_i are the centers and standard deviations of each kernel, respectively.

Learning the weights w_i from a demonstration is a least-squares problem, which can be solved with a variety of methods, e.g. Locally Weighted Linear Regression. Here we adopt the approach of Kulvicius et al. [13] and use the simple delta-rule from artificial neural networks:

$$\Delta w_i^j = \mu \left(y_d(k) - y^j(k)\right), \tag{9}$$

where $j = 1, \ldots, L$, with L being the number of learning iterations, $y_d(k)$, the desired trajectory and $y^j(k)$, the current trajectory at the jth learning iteration. Here, k is the time-index of the i^{th} Gaussian, where $0 \leq k \leq T$. The *learning rate* can be adjusted with parameter μ.

B. Blending between DMPs

Our main reason for using Kulvicius et al.'s parameterization of DMPs is that it allows for easy blending between primitives, preserving smooth velocity profiles and avoiding slowdown or position inaccuracy at the joining point.

For this, a new overlapping set of centers, c', widths, σ', and weights, w' are defined. In the following equations, $i = 1, \ldots, n$, where n is the number of kernels per DMP and $j = 1, \ldots, m$, where m is the number of DMPs.

Centers are re-spaced evenly based on the new combined DMP time, $T' = \sum_{k=1}^{m} T_k$, according to:

$$c_j^i = \begin{cases} cl\frac{T_1(i-1)}{T'(n-1)}, & \text{if} \quad j=1 \\ \frac{T_j(i-1)}{T'(n-1)} + \frac{1}{T'}\sum_{k=1}^{j-1} T_k, & \text{otherwise.} \end{cases} \tag{10}$$

Basis function width is re-scaled following:

$$\overline{\sigma}_j^i = \frac{\sigma_j^i T_j}{T'}. \tag{11}$$

Likewise, the individual weights of each DMP are combined into a single vector.

The goal function is adapted as follows:

$$\tau\dot{r}' = \begin{cases} \frac{\Delta t}{T_j}\left(g_j - s_j\right), & \text{if} \quad \sum_{k=1}^{j-1} T_k \le t \le \sum_{k=1}^{j} T_k \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

C. Goal Generalization

One of the main advantages of Dynamic Movement Primitives compared to other learning from demonstration methods is the inherent ability to generalize a primitive to a new context by changing the goal or start states, without having to reteach the task.

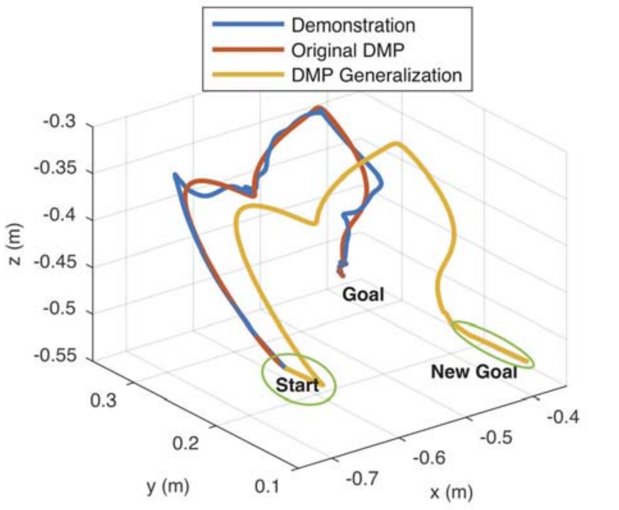

Fig. 1. Sliding during the pickup and placement of the workpiece when generalizing to a new goal.

For Cartesian-space DMPs, it is common to assume that all degrees of freedom are independent of each other. This, however, poses some problems when it comes to generalization in practical scenarios. If the original demonstration recording includes segments of no movement, e.g. when opening/closing a gripper during workpiece pick-up/placement, this will result in unwanted distortion or sliding when generalizing.

In fig. 1, the original trajectory (blue) is first encoded as a DMP (red). It is then generalized to a new goal (yellow). Notice how this causes horizontal sliding along the y-axis during the first and last portions of the movement (green highlights). Figure 2 shows the same trajectory viewed component by component. Ideally, the y-component of the yellow (generalized) trajectory would follow the red and blue from $t = 0$ to $t \approx 6$, and would be parallel to the red and blue trajectories from $t \approx 15$ to the end. We propose to segment the DMP trajectories during gripper activations [20] and during the detected insertion to avoid this issue. The individual segments for pickup and insertion can then be translated, avoiding the sliding seen in figs. 1 and 2.

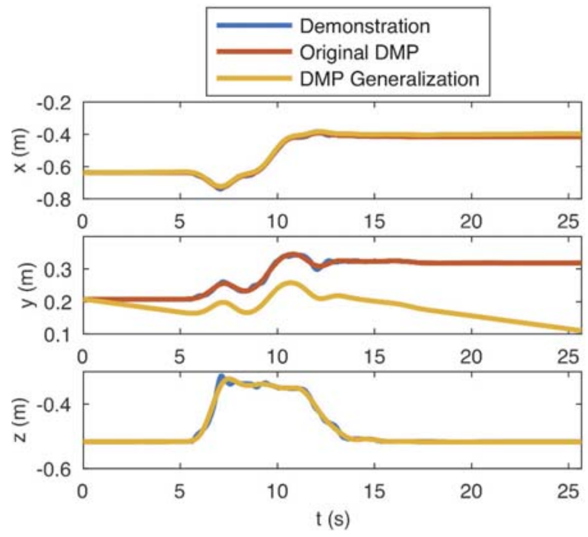

Fig. 2. Notice that in the original demonstration and DMP there are segments of no movement from $t = 0$ to $t \approx 6$ and from $t \approx 15$ to the end. When generalizing to a new goal, movement of the y-axis begins too early, resulting in the sliding shown in fig. 1.

D. Force-Control with DMPs

To avoid exceeding insertion forces that could damage the workpiece, a force control loop at the output of the DMP compensates for contacts during execution. Where consistent force contacts are relevant to the task, it is possible to encode these forces as part of the DMP and have the robot match them during execution [6]. However, for a simple insertion, minimizing contact forces is sufficient.

We thus make use of a simple P-controller on the force-torque sensor input:

$$\mathbf{y}_{SP}(x) = \mathbf{y}_{DMP}(x) + \mathbf{K}_F \cdot \mathbf{q}(x) \cdot \{0, \mathbf{F}\} \cdot \mathbf{q}^{-1}(\mathbf{x}), \tag{13}$$

$$\mathbf{q}_{SP}(x) = \mathbf{q}_{DMP}(x) + \mathbf{K}_M \cdot \mathbf{q}(x) \cdot \{0, \mathbf{M}\} \cdot \mathbf{q}^{-1}(\mathbf{x}), \tag{14}$$

where $\mathbf{y}_{SP}$ and $\mathbf{q}_{SP}$ are the positional and rotational setpoints, and $\mathbf{y}_{DMP}$ and $\mathbf{q}_{DMP}$ are the result of integrating eq. (2) and eq. (5), $\mathbf{K_F}$ and $\mathbf{K_M}$ are force and torque controller gains, $\mathbf{F}$ and $\mathbf{M}$ are the measured forces and torques, adjusted to the end-effector orientation, $\mathbf{q}(x)$.

For our experiments, $\mathbf{K}_F$ and $\mathbf{K}_M$ were adjusted empirically for stability during the insertion.

IV. Convolutional Neural Networks

A. *Using CNNs to find insertion task in demonstrations*

We opted for CNNs to classify temporal data collected during a task demonstration. Since the same sensor is used for robot manual guidance using admittance control and to collect data for task classification, the dynamics of the demonstration and the task are mixed. However, as has been shown in previous work, CNNs are able to find the right features to extract to distinguish an insertion from other movements [18]. Here we applied a similar methodology by training our network with data from tasks demonstrated using admittance control. The collected demonstration data contains n samples of 9 signals that are generated during teaching: the forces, $F_{x,y,z}$, and torques, $T_{x,y,z}$, applied on the end-effector and its velocities in Cartesian space, $V_{x,y,z}$. To input this demonstration data into the CNN, we subsample the demonstration into windows of 1000 time-steps (8 seconds) to generate a fixed-length input. These windows are taken on every demonstration by sliding a window of length 1000 with a stride of 10 on the signals.

With this method, it is only necessary to train the network once with a database of sample demonstrations of insertion tasks. After this, the network can automatically segment future demonstrations, without any additional training.

1) Insertion detection and localization using demonstration F/T signals: To determine if an input $X^{1000\times9}$ contains an insertion task demonstration, a VGGnet-like [21] network was used. Figure 3 shows the chosen architecture. The flow of layers is separated into blocks, where each block contains 2 layers of 1-dimensional convolution (with n 3x1 filters, for time-based data) followed by a max-pooling layer. Batch normalization is used after every convolution layer to speed up the training. For each of these convolution blocks, a set amount of 3x1 filters is applied and this number is increased for the subsequent block. By doing so, the pooling blocks decrease the size of the input at each layer, but the increasing number of filters makes the number of trained features higher. Three convolution blocks are used to learn the low-level features of the signal and are shared between the classification and localization parts of the network.

After the third convolution block, a specialized convolution block is trained to learn features that are important to classify insertion signals (yellow section of figure 3). The remaining feature maps are fed into two softmax neurons in a fully-connected method to get a binary classification of the input signal.

In addition to being able to detect if a segment of signal contains an insertion task, we also need to be able to localize the time when the insertion was performed. This is done by a specialized convolution block trained to learn features that are important to localize insertion tasks (blue section of figure 3). Since the localization part of the system is not a binary output, we used two densely connected neurons with linear activation to represent the beginning time and end time of the insertion task in the input signal.

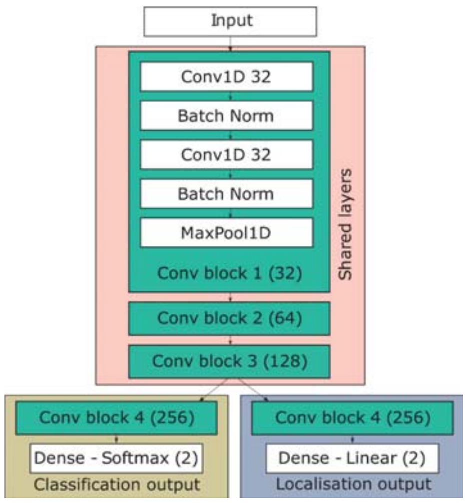

Fig. 3. The CNN architecture used for insertion detection and localization. Each convolution block contains two 1D convolution layers with n filters of size 3 followed by batch normalization layers and a max-pooling of size 2 with a stride of 2. The numbers in parenthesis indicate the number of filters used for the convolution layers of that block. The input layer is 1000x9, which correspond to 8 seconds of demonstration F/T signals.

2) Training the network for classification and regression: As mentioned in section IV-A.1, every inference of an input will be assigned a binary class (*no event* or *insertion task*) and a position of the occurrence of the insertion. This multi-task network implies that we need to minimize two cost functions during training: a cross-entropy for classification, and a mean squared error loss for regression.

The classification of segments was trained first by taking segments of 8 seconds (the network input size) and labeling them positively or negatively based on the presence of an insertion task. To be a positive segment, the window of 8 seconds needs to overlap over 70% of the ground-truth insertion segment and to be a negative segment, the window needs to overlap less then 30% of the ground-truth. Segments overlapping an insertion task with a ratio between those two thresholds were not used for training. This is expressed in eq. (15).

$$y_{gt} = \begin{cases} 0, & r_i \leq 0.3 \\ 1, & r_i \geq 0.7 \\ undefined, & 0.3 < r_i < 0.7. \end{cases} \tag{15}$$

Here, y_{gt} is the ground-truth classification of a demonstration segment and r_i is the ratio of the annotated region of insertion in the original demonstration that is covered by the segment window.

B. *Task Segmentation with Convolutional Neural Networks*

1) Generating the insertion segment based on the original demonstration: To get the predicted insertion task segment in a complete robot demonstration, we first need to rearrange the demonstration data in a way that fits the CNN's input size. This is done in a similar manner as described in section IV-A.

Inference is done using these n-generated windows on the trained CNN and windows classified as insertion task with a confidence level higher than 95% are kept in order to determine the insertion location.

The start and end index determined during the inference of the positive windows are used to generate the insertion segment in the original demonstration. Points generated by a window $X_{\{n,n+1000\}}$ can be mapped to the original demonstration index by adding n to the points index, where n is the start index of the window on the complete demonstration.

2) Generating the approach point based on the insertion segment: Once the insertion segment position has been determined, we locate the approach point of the insertion task. This is done naively by taking the robot position at the insertion and retreating a certain distance along the approach path in the demonstration data.

We took the position of the end-point index as reference since the detected position of the end point is more reliable than that of the start point, as covered in section VI-A. The approach point is then determined by retreating 50mm along the trajectory path, using point-to-point Euclidean distance as the measure. Alternatively, we could have chosen to retreat a set distance vertically, instead of along the demonstration trajectory, and then modify the approach DMP accordingly. It could also be chosen based on demonstration velocity, since demonstrators will typically slow down during the approach to achieve greater accuracy.

3) Using segmentations with DMPs: Once the approach and insertion indices have been determined, these are used, in combination with gripper activation indices, to segment the original demonstration file into: initial approach – pickup – insertion approach – insertion – retraction.

We learn separate DMPs for each segments, and combine them into a single full-task DMP, as described in section III-B. We can thus modify individual segments, granting better control of task execution and generalization.

We can, e.g. locally speed up the approach, pickup and retraction motions while slowing down the insertion to reduce contact forces. We can apply force control during the insertion, or modify the approach online for, e.g., obstacle avoidance, while keeping the insertion action unperturbed.

V. Experimental Setup & Procedure

A. Robot Setup

1) Hardware: We used a Universal Robots UR5 robot arm with a Robotiq FT 300 force-torque sensor and a Robotiq 2-Finger 85mm robot gripper for our experiments

2) Software: The UR5 was running software version 3.5.4. The robot demonstration recording, DMP learning and control code was implemented in Java as a software extension and bundled as a URCap (Universal Robots software extension package). The CNN was implemented in Keras [22] with TensorFlow backend on Python 2.7, and was run on a separate PC, connected to the robot over a LAN. The URCap sent demonstration files to the CNN PC, which returned the segmented task that was then encoded as a set of DMPs.

B. Workpieces and Assemblies

In our experiments we wished to evaluate the performance of our learning from demonstration framework for workpieces with tolerances similar to industrial assembly scenarios.

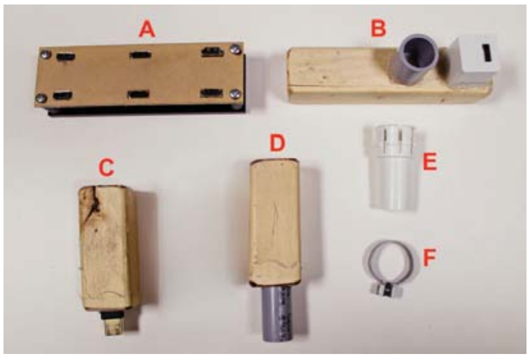

Fig. 4. Workpieces used for the assembly experiments: USB female insertion rig (A), USB and Pipe female insertion rig (B), USB male peg (C), pipe (D), housing shaft (E) and retainer ring (F).

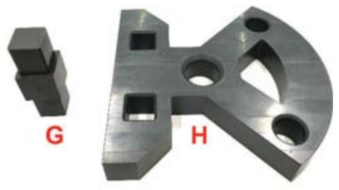

Fig. 5. Subset of the Cranfield assembly benchmark, including the square peg (G) and backplate (H).

Figures 4 and 5 show the workpieces used for our assembly experiments. The USB (C) peg is to be inserted into its respective female connector in (A) or (B). The pipe (D) is to be inserted into (B). The retainer ring (F) is to be clicked into place on top of the housing shaft (E). The Cranfield square peg (G) is inserted into one of the two square holes in the backplate (H). All of the pegs (C, D) have been fitted into wooden blocks with a square profile to facilitate grasping by the robot gripper and eliminate variance during gripping, since this is not a part of our experiments.

C. Procedure

1) Experiment 1: Detection of Insertion: To test the CNN's efficiency at segmenting insertion tasks, we built a training set containing 1008 complete demonstrations where insertion, pick-and-place and various other tasks were performed. Multiple objects were used to perform these demonstrations, including those mentioned in section V-B.

After the training phase, 58 other demonstrations containing insertion and pick-and-place tasks were recorded to be used as a test set. The performance of the CNN was evaluated on this test set by measuring the detection rate and the precision of the localizations of tasks.

2) Experiment 2: Generalization Performance: To evaluate the advantages of segmentation for DMP generalization, we taught the robot to insert a male USB peg into a female USB insertion rig. After learning the task, we replayed it while moving the insertion location in increments of 5 mm along the x- and y-axes in the robot coordinate frame, and compared performance for regular and segmented DMPs. For segmented DMPs, only the insertion segment goal was generalized to the new insertion location.

3) Experiment 3: Assembly Tasks: For each of the pieces shown in figs. 4 and 5, we demonstrated an insertion task and learned a segmented DMP. We replayed the task three times. We taught a new goal (I) with a change in position and replayed this three times. The position of this new goal (I) was taught again two more times, for a total of three teachings and nine replays of the same goal, to test both DMP replay robustness and robustness to different ways of teaching. The same sequence was then repeated for another goal (II) with a change in both position and orientation. The procedure can be observed in fig. 6.

VI. Experimental Results

A. Insertion Task Detection Performance

As data from a demonstration is fed to the proposed CNN, if an insertion task is detected, the system will output a segment of time where it predicted the event occurred. In this experiment, a prediction is considered correct if the intersection of the predicted segment and the ground truth over the total ground truth segment is at higher than 50%.

Using the test set of 58 demonstrations, we obtained the results in table I. We can observe a recall rate of 94.7%, i.e. 55 out of the 58 insertion tasks in the test set demonstrations were located and classified correctly. Out of those 3 errors, one was correctly detected as an insertion task, but the segment time was incorrectly reported.

TABLE I
Insertion task detection results in user demonstrations.

	Predicted positive	Predicted negative
Actual positive	55	3
Actual negative	7	-

Recall rate	94.7%
Precision rate	88.7%

We obtained a precision rate of 88.7%, meaning that some predicted insertion tasks regions were false positives. Of these, most were targeting a sequence in the demonstration where the operator was putting down an object on the workbench. While detecting this as an insertion task is not a desired behavior, this type of task is very similar in motion to an insertion (putting down an object versus inserting down an object) and the signals generated during the demonstration are also similar, explaining the false positive predictions.

B. Generalization Performance with and without Segmentation

Figure 7 shows the Cartesian space trajectories of the robot for the original goal, and increments of 5 - 15 mm away from it, for both regular and segmented DMPs. Regular DMPs result in a collision and a task failure, because they stay too close to the original trajectory throughout most of the movement and the generalization happens only in the last stages, close to the insertion.

C. Task Success Rates

The task success rates for each of the four workpieces and conditions can be seen in table II. For the original task, the success rate is universally 100%. Generalizing to a new position drops the overall success rate for all pieces to 79.63%, but there is a significant variation between the different pieces. Both the square peg and PVC tube have high success rates (81.48% and 96.3%, respectively). However, particularly for the case of the metal ring, the rate drops to 62.96%. Generalizing to a new orientation results in much poorer performance. Here, the average success rate for all pieces is 55.14%, where the maximum is 70.37% for the square peg, but the minimum is 25.93% for the metal ring.

VII. Discussion

A. Performance & Results

As seen in section VI-B, segmentation overcomes the issue of shape distortion when generalizing sigmoidal DMPs to new goals. By preserving the shape of the insertion segment from the original DMP and translating it, we maintain the same insertion trajectory, regardless of the distance between the original and new goals.

We observe that applying the DMP to the workspace configuration taught in the original demonstration achieves 100% success almost universally. This performance is therefore robust enough for the industry. The same cannot be said of the generalization results (table II). We found they were greatly affected by the quality of the demonstration and how the new goal was taught. Inherently hard to demonstrate tasks, e.g. the metal ring, performed worse under generalization. The orientation of the new goal had a large effect on success or failure. The use of a general purpose gripper resulted in a degree of compliance in the workpiece grasp. Thus, a particular workpiece orientation was not constrained to a single robot position, and could slide in the gripper. Due to this, if the new goal orientation was crooked in comparison to the original goal, the task execution was more likely to fail. This can be seen in the results of the metal ring, where the initial task demonstration had to be repeated several times due to its complexity.

Even though task success is greatly affected by the quality of teaching, DMPs for motion control are robust in themselves, as we found that the results were consistent from one task execution to another. The only exceptions were pieces with tight tolerances, e.g. the metal ring and USB, in cases where the demonstration was already close to failure, such that extremely small compliances could lead to task failure,

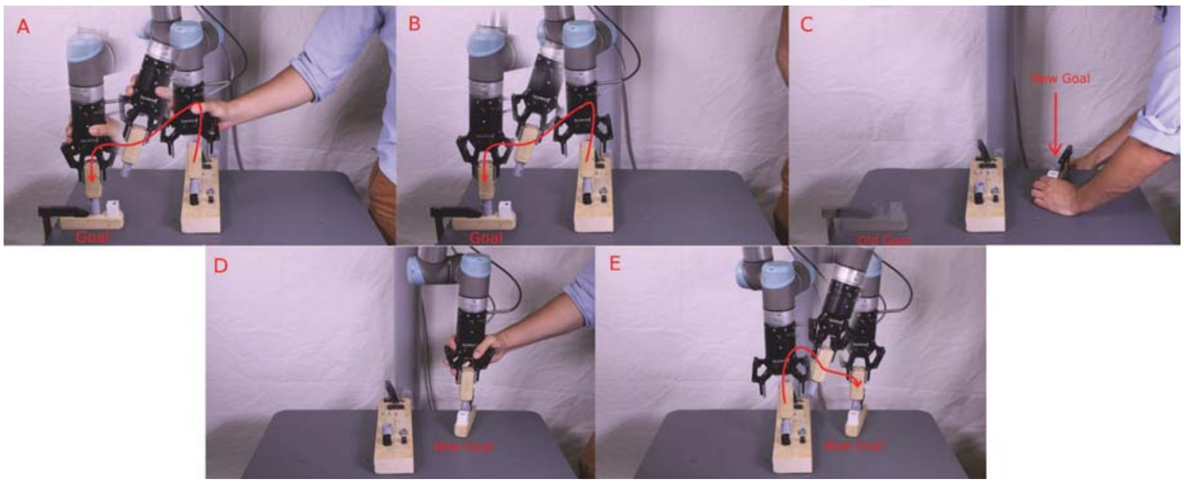

Fig. 6. Experiment 3 procedures: (A) Teach an insertion task and generate the segmented DMP path. (B) Replay the generated task 3 times. (C) Move the goal point to a new position. (D) Teach new goal without regenerating the DMP. (E) Replay the new goal task 3 times. (F) Repeat C-E for another goal position.

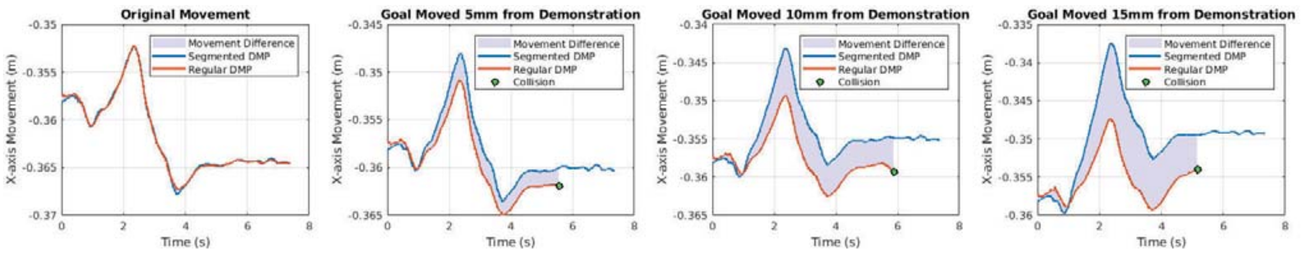

Fig. 7. Comparison of regular and segmented DMP movements as the goal is moved away from the original in increments of 5mm in the x-axis. Notice that the paths deviate early in the movement and that the regular (non-segmented) DMP fails already when the goal is moved 5mm away from the original.

TABLE II

SUCCESS RATES FOR EACH OF THE WORKPIECES UNDER EACH OF THE THREE CONDITIONS: ORIGINAL GOAL (SAME AS DEMONSTRATED), NEW GOAL (I) WHERE ONLY THE POSITION IS CHANGED, AND NEW GOAL (II) WHERE BOTH POSITION AND ROTATION ARE CHANGED.

	Square Peg	Metal Ring	USB	PVC Tube	All Pieces
Original Goal	1	1	1	1	1
Position Change	0.8148	0.6296	0.7778	0.963	0.7963
Position & Rotation Change	0.7037	0.2593	0.5769	0.6667	0.5514
Total	0.7937	0.5238	0.7258	0.8413	0.7211

Note that our method outperforms Gao et al. [3] when it comes to replaying the original task, as, except for the square peg, all of our assemblies are transition fits – with the PVC tube being the tightest, as tight-to-similar fit – while their tightest fit is of the location clearance (H7/h6) type, and our average success rate for all pieces is 100%, while theirs is 88%. Additionally, we learn from a single demonstration, while their method (GMM) requires several. However, we cannot directly compare our generalization results to theirs, as they simply generalize from tighter to looser tolerances, whereas we generalize the same insertion to different workspace positions and orientations.

We have to point out that, when generalizing to a new location, some of our successful executions would not be ideal for production. In the PVC tube assembly the tube hit the rim of the receiver with considerable force before sliding in. This could damage the finish of production workpieces.

B. Future Work & Considerations

We believe that future work should consider approaches that do not mimic the user's demonstration, but instead replace each segment with a strategy or algorithm that is more robust and adaptable.

Our results show that demonstrations can be unreliable, as insertion trajectories are far from optimal and, due to compliance, generalization performance depends on the relative difference in orientation between the original and new goal configurations. While DMPs can adapt to the contour of the insertion using force-torque feedback, we currently do not consider demonstration forces. Using the CNN, we could incorporate force-torque feedback strategies for locating the hole. This would make the approach more robust to workpiece misalignments and also address the inaccuracies in the demonstration, since the search technique would avoid collisions at the insertion point. It would also allow us to optimize each of the trajectory segments individually.

VIII. Conclusion

We have presented a method to learn robot movement primitives suited for industrial assembly tasks, specifically insertions, where high accuracy is required. The task is taught to a robot by guiding it using admittance control. A Convolutional Neural Network detects and segments the insertion based on contact forces captured by a force-torque sensor. This, combined with gripper activations divides the task into phases that are encoded as Dynamic Movement Primitives. DMPs allow us to perform one-shot learning, and provide a real-time adaptable movement encoding that can be generalized to new configurations and combined with a force controller to reduce insertion forces. In our experiments we have shown that our approach is more robust to generalization than regular unsegmented DMPs. However, due to imperfections in the users' demonstrations and relative orientation differences in the new goal configurations caused by grasp compliance, task replays can still fail when generalizing to new positions and orientations. Future approaches should concentrate on utilizing high-level information extracted from the CNN to optimize the insertion beyond the user demonstration by using force-controlled search strategies, which can be previously learned or selected by the user from a library.

References

[1] I. Iturrate, E. H. Østergaard, M. Rytter, and T. R. Savarimuthu, "Learning and correcting robot trajectory keypoints from a single demonstration," in *Proceedings of the International Conference on Control, Automation and Robotics (ICCAR)*, pp. 52–59, IEEE, 2017.

[2] D. Massa, M. Callegari, and C. Cristalli, "Manual guidance for industrial robot programming," *Industrial Robot: An International Journal*, vol. 42, no. 5, pp. 457–465, 2015.

[3] X. Gao, J. Ling, X. Xiao, and M. Li, "Learning force-relevant skills from human demonstration," *Complexity*, vol. 2019, 2019.

[4] Robotiq Inc., "Force copilot," 2018. Accessed on: *https://robotiq.com/products/force-copilot* on 2019-02-19.

[5] E. Shahriari, A. Kramberger, A. Gams, A. Ude, and S. Haddadin, "Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement primitives," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, pp. 136–142, 2017.

[6] B. Nemec, F. J. Abu-Dakka, B. Ridge, A. Ude, J. A. Jorgensen, T. R. Savarimuthu, J. Jouffroy, H. G. Petersen, and N. Kruger, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pp. 1–7, IEEE, 2013.

[7] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1398–1403, 2002.

[8] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

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

[10] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 91–98, 2008.

[11] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 365–371, 2011.

[12] B. Nemec and A. Ude, "Action sequencing using dynamic movement primitives," *Robotica*, vol. 30, no. 5, pp. 837–846, 2012.

[13] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[14] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, pp. 1096–1104, 2009.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[16] H.-I. Lin and Y. Chiang, "Understanding human hand gestures for learning robot pick-and-place tasks," *International Journal of Advanced Robotic Systems*, vol. 12, no. 5, p. 49, 2015.

[17] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2017.

[18] E. Roberge and V. Duchaine, "Detecting insertion tasks using convolutional neural networks during robot teaching-by-demonstration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3210–3216, 2017.

[19] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH Computer Graphics*, vol. 19, pp. 245–254, 1985.

[20] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction," *Autonomous Robots*, pp. 1–17, 2018.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[22] F. Chollet *et al.*, "Keras." https://keras.io, 2015.