

Combining Programming by Demonstration with Path Optimization and Local Replanning to Facilitate the Execution of Assembly Tasks

1st Gal Gorjup

*Dept. of Mechanical Eng.
The University of Auckland
Auckland, New Zealand
ggor290@aucklanduni.ac.nz*

2nd George P. Kontoudis

*Bradley Dept. of Electrical and Computer Eng.
Virginia Tech
Blacksburg, USA
gpkont@vt.edu*

3rd Anany Dwivedi

*Dept. of Mechanical Eng.
The University of Auckland
Auckland, New Zealand
adwi592@aucklanduni.ac.nz*

4th Geng Gao

*Dept. of Mechanical Eng.
The University of Auckland
Auckland, New Zealand
ggao102@aucklanduni.ac.nz*

5th Saori Matsunaga

*Information Technology R&D Center
Mitsubishi Electric Corporation
Tokyo, Japan*

6th Toshisada Mariyama

*Information Technology R&D Center
Mitsubishi Electric Corporation
Tokyo, Japan*

matsunaga.saori@dc.mitsubishielectric.co.jp mariyama.toshisada@dc.mitsubishielectric.co.jp

7th Bruce MacDonald

*Centre for Automation and Robotic Eng. Science
The University of Auckland
Auckland, New Zealand
b.macdonald@auckland.ac.nz*

8th Minas Liarokapis

*Dept. of Mechanical Eng.
The University of Auckland
Auckland, New Zealand
minas.liarokapis@auckland.ac.nz*

Abstract—With the emergence of agile manufacturing in highly automated industrial environments, the demand for efficient robot adaptation to dynamic task requirements is increasing. For assembly tasks in particular, classic robot programming methods tend to be rather time intensive. Thus, effectively responding to rapid production changes requires faster and more intuitive robot teaching approaches. This work focuses on combining programming by demonstration with path optimization and local replanning methods to allow for fast and intuitive programming of assembly tasks that requires minimal user expertise. Two demonstration approaches have been developed and integrated in the framework, one that relies on human to robot motion mapping (teleoperation based approach) and a kinesthetic teaching method. The two approaches have been compared with the classic, pendant based teaching. The framework optimizes the demonstrated robot trajectories with respect to the detected obstacle space and the provided task specifications and goals. The framework has also been designed to employ a local replanning scheme that adjusts the optimized robot path based on online feedback from the camera-based perception system, ensuring collision-free navigation and the execution of critical assembly motions. The efficiency of the methods has been validated through a series of experiments involving the execution of assembly tasks. Extensive comparisons of the different demonstration methods have been performed and the approaches have been evaluated in terms of teaching time, ease of use, and path length.

Index Terms—flexible manufacturing, assembly, programming by demonstration, system integration

The funding for this research was obtained by Mitsubishi Electric Corporation under UniServices grant agreement #5000736.

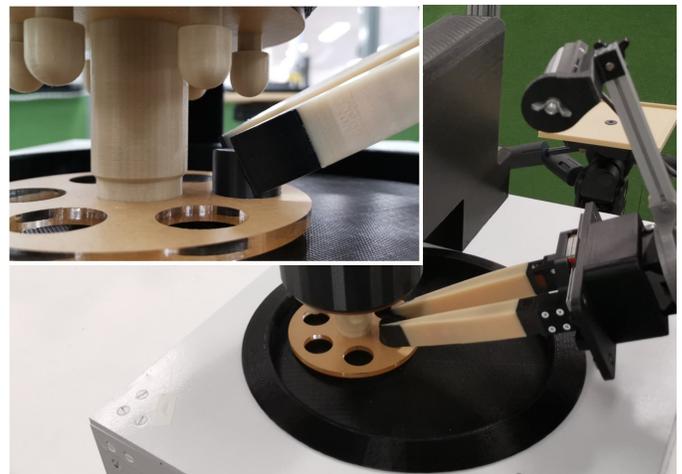


Fig. 1. The robot arm and gripper system performing a peg-in-hole task on a polishing machine replica. The insertion is executed in a heavily constrained space, which corresponds to real-life assembly scenarios.

I. INTRODUCTION

Industrial robots have allowed production and assembly lines to reach extraordinary levels of automation, increasing process efficiency and minimizing faults. Adapting to the high rate of technological advancement, the market has started to move away from mass production, demanding progressively

more customized products. In order to effectively respond to the rapidly changing task requirements, robot systems must allow for fast and efficient re-programming that does not require expert knowledge in the field. Traditional robot programming methods (such as pendant teaching) have proven to be less efficient in cases where the system must be adapted to novel, complex tasks on a regular basis. Online, pendant-based teaching methods are often tedious, time-consuming, and introduce production delays because the robot must be halted for the duration of re-programming. Offline, simulation-based methods can be utilized without using the physical robot, but require professional knowledge and a considerable amount of effort to be properly configured. These issues have motivated several industrial and research groups to develop alternative methods of transferring human skill to robot systems.

Most skill transfer approaches are based on Programming by Demonstration (PbD), where the human operator demonstrates a task or skill, which is then transferred to the target robot system. The skill transfer can be arbitrarily complex, ranging from simple trajectory reproduction to sophisticated skill acquisition that interprets the task context and adapts to external disturbances. Teaching by showing comes naturally to human operators, making most developed PbD approaches accessible to untrained operators. From the user perspective, the time and effort required to program tasks of varying complexity is fairly low when using an effective PbD framework. The flexibility offered by such approaches facilitates the development of reconfigurable manufacturing environments suited for the future. In addition, it introduces new automation frontiers for smaller businesses that may require the robots to perform short term, daily changing tasks. The concept of PbD can also play a major role in human-robot collaboration, where the robot can learn and adapt its behavior based on human observation.

This paper presents a human to robot skill transfer methodology that combines programming by demonstration with path optimization and local replanning to allow for the efficient execution of assembly tasks. The methodology performs collision-aware optimization of non-functional demonstrated trajectory segments in terms of path length, ensuring fast execution of the desired task. This reduces the effects of sub-optimal user demonstrations that may influence the quality of programmed motions in other, learning based approaches [1]. The system can adapt to dynamic task goal changes by employing a local replanning scheme that updates the tail of the global path during its execution, based on real-time feedback from the perception system. The efficiency of the proposed methods has been validated with a series of experiments focusing on the execution of peg-in-hole assembly tasks with a robot arm and gripper system. As a secondary contribution, three different task demonstration approaches have been compared and evaluated in terms of teaching time and ease-of-use.

The rest of the paper is organized as follows: Section II introduces the related work, Section III describes the utilized apparatus, Section IV presents the developed framework, Section V presents and discusses the results, while Section VI concludes the paper.

II. RELATED WORK

This section provides a brief literature review of the fields relevant to this work. This includes related work in PbD, trajectory optimization, and path planning.

A. Programming by Demonstration

Over the last decades, significant research effort has been invested in the field of PbD, with developments ranging from simple trajectory mapping to high level representations of complete task procedures [1]–[3]. Such approaches have also been employed in robotic assembly studies [4], where much of the work has focused on a set of representative tasks, such as peg-in-hole, pick-and-place, bolt screwing, etc.

Several methods of capturing user demonstrations have been considered in literature. Kinesthetic teaching and teleoperation produce data that is recorded directly on the target robotic platform with the human operator compensating for kinematic and dynamic embodiment differences. This makes the demonstrations easier to interpret, but can sometimes be inconvenient for the operator and prevent them from effectively demonstrating the target skill [5]. Alternatively, a PbD scheme can be based on motions that the operators execute with their own bodies, mapping them to the target platforms [6], [7]. Although this is more intuitive for the operator, it requires additional human to robot motion mapping methods [8].

Once the demonstration trajectories have been obtained, the simplest approach is to map them directly to the robot platform. While this might be sufficient for simple motions with loose accuracy requirements, such an approach is typically inadequate for assembly oriented applications. Such a basic motion replay cannot account for external disturbances and any operator errors are propagated to the robot. These errors can be amended either through independent trajectory optimization or by employing appropriate local replanning methods.

B. Trajectory Optimization

In assembly tasks, the operator is typically trying to program short, efficient robot motions that complete a given task as quickly as possible. Human demonstrations are in most cases sub-optimal in terms of speed of execution. To effectively shorten the path length, the human demonstrated trajectories can be efficiently optimized by employing appropriate path planning algorithms [9]. Moreover, given an existing robot trajectory, methods such as the Shortcut or the Partial Shortcut [10], are able to refine and shorten the path with respect to the identified obstacles in the space. Alternatively, reinforcement learning methods that produce a set of feasible trajectories in constrained environments can also be employed [11]. As such methods discard any functional motion that may be present in the segment, they can only be applied to the non-functional parts of the demonstrated skill. In this context, functional motions refer to trajectory segments and force profiles that are critical for the success of a particular assembly task, such as part alignment and insertion. Non-functional motions refer to trajectory segments that can be significantly altered without affecting task success, such as the reach to grasp path.

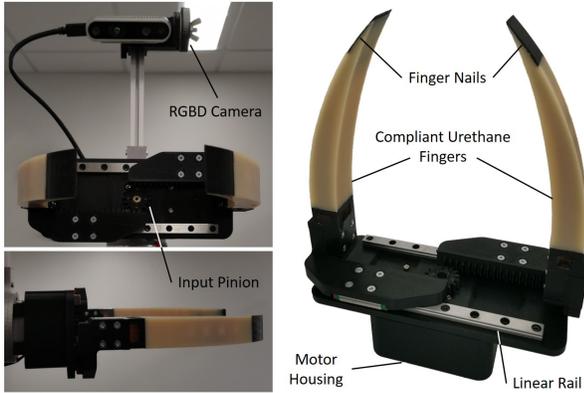


Fig. 2. The employed parallel-jaw gripper with the pre-shaped adaptive robot fingers. A single actuator is used to control both fingers, which move in a coupled manner on linear rails. Upon contact with the object, the pre-shaped fingers conform to the object shape. An off-centered mount houses the Intel RealSense RGBD camera.

C. Path Planning and Local Replanning

The path planning problem for continuous obstacle spaces and arbitrary robot shapes has been addressed by sampling-based algorithms, such as the rapidly-exploring random tree (RRT) [12] and the probabilistic roadmaps (PRM) [13]. A variant of the RRT with rewiring of the search-tree, the RRT*, was proposed in [14]. RRT* is a probabilistically complete and asymptotically optimal, single-query algorithm that can be applied only to static environments. Replanning can be employed to efficiently identify a new path for dynamic or even uncertain environments. An asymptotically optimal sampling-based, single-query framework for dynamic environments, namely RRT^X, was presented in [15]. The methodology performs local replanning to repair and refine the precomputed global search-graph. The authors in [16] introduced the informed RRT* that significantly reduces the convergence rate of RRT*. The latter instead of employing the sampling-based algorithm in the whole space, it reduces the problem to a relatively smaller space (prolate hyperspheroid) and dynamically compresses the local space based on the gathered information. In [17], local replanning was utilized to recompute a path in an online fashion. In this approach, the RRT* has been executed on a relatively small space, where motion planning *completeness* is guaranteed with topological connectedness tools.

III. APPARATUS

In this section, the employed apparatus is presented. The robot platform is based on a 6 Degrees of Freedom (DoF) serial manipulator with a maximum payload of 5 kg. The gripper used is a parallel-jaw design with adaptive, pre-shaped fingers [18] (see Fig. 2). The gripper is chosen as it maximizes the contact area with the objects while grasping and compensates for object positioning uncertainties. The perception module relies on an Intel RealSense depth camera (model D435) mounted on the robot end-effector. The hand-eye calibration is performed through fiducial marker detection, as described in [19].

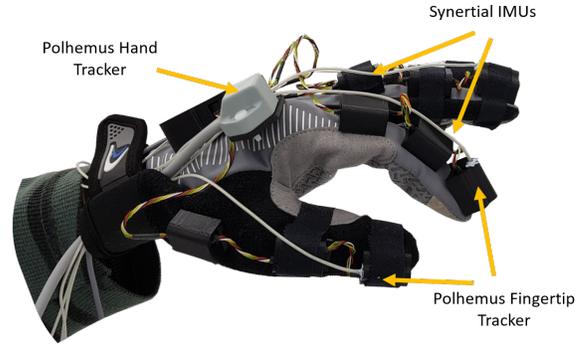


Fig. 3. The proposed dataglove system that comprises of IMU and magnetic motion capture sensors.

The system was experimentally validated with a peg-in-hole task executed on a replica of the Rana-3 sample polishing machine (Fig. 1). The task involves inserting cylindrical specimens into the carrier disk of the polishing machine, which stops at an arbitrary angle after every polishing cycle. The cylindrical samples have varying height and a diameter of $\text{Ø}29.5 \pm 0.1$ mm, while the hole diameter is $\text{Ø}30.2 \pm 0.1$ mm. Since the samples are circular, this insertion task is comparably easier than one involving prismatic shapes. Even though this is not evaluated in this work, the in-hand object pose tracking component of the framework is designed to account for such object pose considerations.

A. Dataglove

To record human demonstrations, a dataglove combining an inertial and a magnetic motion capture system was developed (Fig. 3). The Synertial Cobra sensors were selected as the inertial motion capture system. More precisely, 16 Inertial Measurement Units (IMUs) were used per hand and two additional IMUs were used for the forearm and the upper-arm. The Cobra system does not suffer from occlusion-related issues and is reasonably robust to magnetic disturbances. However, since it is IMU-based, the system accuracy is limited and the measured angles tend to drift over time. These errors propagate through the skeleton to the fingertips. Furthermore, the inertial glove is not able to measure the hand pose with respect to the global coordinate frame and a system with absolute pose estimation capability is required. To overcome the drift and absolute reference issues, a magnetic motion capture system was included to track the fingertips and correct the inertial system data. For this purpose, the Polhemus Liberty magnetic motion capture system was chosen. The Liberty system offers accurate pose tracking through standard sensors that can be attached to the hand or arm, and micro sensors that can be mounted on the fingertips (Fig. 3). The measurements are expressed in the form of transformation matrices with respect to the global reference frame and are therefore not prone to drifting, making the system appropriate for correcting the IMU-based system. The proposed dataglove thus combines in a synergistic manner IMU and magnetic motion capture sensors to provide accurate and robust measurements.

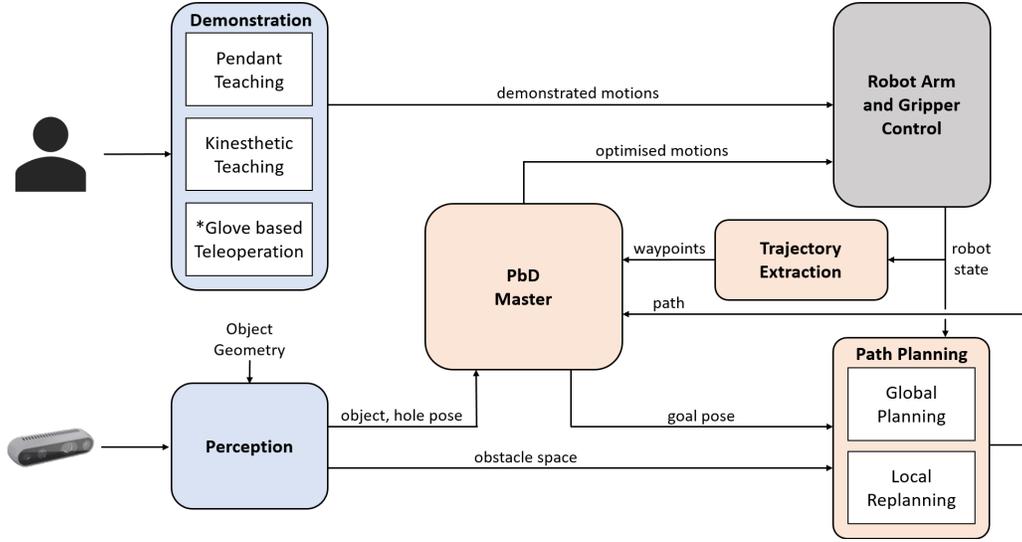


Fig. 4. Framework architecture for the proposed Programming by Demonstration methodology. Pendant and kinesthetic teaching are executed manually. The glove based teleoperation involves a human to robot motion mapping procedure that projects the demonstrated motions onto the robot joint space.

B. Trajectory Optimization

The correction of the joint angles from the inertial motion capture system is performed by solving the inverse kinematics as a constrained numerical optimization problem, where the goal pose is the Liberty pose estimate for each finger. The decision variables are the joint angles, and the non-equality constraints are determined by the joint limits. The same formulation is used for all five fingers of the hand and the combined results provide the calibrated joint values of the full human arm hand system. Let \mathbf{R} represent the current rotation matrix and \mathbf{R}_{goal} the goal rotation matrix, then the angle difference $\Delta\theta$ can be obtained by,

$$\Delta\theta = \arccos\left(\frac{\text{trace}\left(\left(\mathbf{R}\right)^{-1}\mathbf{R}_{\text{goal}}\right) - 1}{2}\right), \quad (1)$$

Furthermore, let $\mathbf{x} = f(\mathbf{q})$ be the forward kinematics mapping from joint to task space for each finger and \mathbf{q} is the vector of joint angles for each finger. Let also $\mathbf{x}_{\text{goal}} \in \mathbb{R}^3$ be the goal translation vector. Then, the objective function for the correction of the joint angles can be defined as,

$$C(\mathbf{q}) := w_x \|\mathbf{x} - \mathbf{x}_{\text{goal}}\| + w_r \Delta\theta + w_s \|\mathbf{q} - \mathbf{q}_{\text{imu}}\|, \quad (2)$$

where w_x and w_r are the position and rotation goal weights and w_s is the weight of the similarity measure and \mathbf{q} is the joint angle vector from the inertial motion capture system. The weights are nonnegative and they sum up to one. Finally, the optimal pose is obtained by,

$$\mathbf{q}^* = \underset{\mathbf{q} \in \mathcal{Q}}{\text{argmin}} C(\mathbf{q}), \quad (3)$$

where $\mathcal{Q} \in (\mathbf{q}^-, \mathbf{q}^+)$ is the feasible joint set bounded by the lower \mathbf{q}^- and upper \mathbf{q}^+ joint limits.

IV. FRAMEWORK

The PbD framework was implemented with the Robot Operating System (ROS) [20], which provided the necessary communication, testing, and visualization utilities. Its architecture is presented in Fig. 4, where the blocks conceptually correspond to the implemented ROS node structure. The *Demonstration* module handles capturing and translating human demonstrations to the robot platform, relying on either kinesthetic teaching or dataglove teleoperation. Trajectories generated from human demonstrations are executed on the physical robot system, which forms the *Robot Platform* module, along with their respective interfaces built with ROS Control [21]. The *Perception* module contains an interface for the Intel RealSense camera that streams color and depth data to the network. The data is filtered by the point cloud processing module, which also handles object tracking and target pose estimation. The module relies on the PCL library [22]. The *Path Planning* module relies on the MoveIt [23] development platform, which maintains a simulated environment based on the robot state and collision space. The global planning and local replanning strategies are based on the RRT* algorithm [14]. The *PbD Master* module represents the central framework component, handling state control and task execution.

A. Demonstration Approaches

Trajectory samples were captured through two demonstration methods: i) kinesthetic teaching and ii) teleoperation based teaching. Kinesthetic teaching was based on the built-in robot gravity compensation mode, with an external trigger for gripper opening and closing. In this setting, the user physically guides the robot, operating the gripper when necessary. In the second approach, hand and finger motion data obtained from the dataglove are mapped to the robot arm and gripper in real time. This mapping allows the user to execute the

Algorithm 1: HoleEstimate ($P_{\text{base}}, \mathbf{T}_{\text{base}}^{\text{target}}, l, \alpha, P_{\text{obj}}, \rho$)

```
1  $\mathbf{T}_{\text{base}}^{\text{hole}} \leftarrow \mathbf{T}_{\text{base}}^{\text{target}}$ ;
2  $P_{\text{target}} \leftarrow \text{TransformCloud}(P_{\text{base}}, \mathbf{T}_{\text{base}}^{\text{target}})$ ;
3  $P_{\text{local}} \leftarrow \text{CropCloud}(P_{\text{target}}, l)$ ;
4 if  $P_{\text{local}} \neq \emptyset$  then
5    $(P_{\text{inliers}}, C_{\text{model}}) \leftarrow \text{SACPlaneSegmentation}(P_{\text{local}})$ ;
6    $P_{\text{plane}} \leftarrow \text{ProjectInliers}(P_{\text{inliers}}, C_{\text{model}})$ ;
7    $P_{\text{polygons}} \leftarrow \text{ConcaveHull}(P_{\text{plane}}, \alpha)$ ;
8    $d_{\text{min}} \leftarrow l$ ;
9   for each  $P \in P_{\text{polygons}}$  do
10    if  $\text{AreaRatio}(P, P_{\text{obj}}) > \rho$  then
11       $d \leftarrow \|\mathbf{t}^{\text{hole}} - \mathbf{t}^P\|$ ;
12      if  $d < d_{\text{min}}$  then
13         $\mathbf{T}_{\text{base}}^{\text{hole}} \leftarrow \mathbf{T}_{\text{base}}^P$ ;
14         $d_{\text{min}} \leftarrow d$ ;
15 return  $\mathbf{T}_{\text{base}}^{\text{hole}}$ .
```

task without physically interacting with the platform. In this case, *translation mapping* is relative to the initial position, while *rotation mapping* is absolute with respect to a common reference frame. Gripper aperture control is based on the distance between the human thumb and index fingertips.

B. Perception

The perception module handles filtering and interpretation of streamed point cloud data for the purposes of obstacle space identification, pose tracking of the grasped object, and target (hole) estimation. A distance limit filter is applied to remove the gripper points from the point cloud set before passing the depth data to the path planning module. Object pose tracking during grasping and manipulation is crucial for the success of most assembly tasks, especially when utilizing non-rigid, adaptive grippers, which are susceptible to grasping inaccuracy and post-contact reconfiguration. The grasped object pose with respect to the end-effector is in this setting tracked through a two-stage, parallel point cloud processing procedure. The first component performs global pose estimation by aligning Fast Point Feature Histogram (FPFH) features [24] of the object mesh to the observed scene. To ensure adequate alignment speed, a pre-rejective variant of the Random Sample Consensus (RANSAC) procedure [25], is used. Once an initial pose estimate is available, continuous, real-time tracking is performed by the Iterative Closest Point (ICP) algorithm [26].

In several applications, including the one examined in this work, the assembly target pose may change between executions. For the peg-in-hole task, an online, vision-based goal pose estimation algorithm was developed, as presented in Algorithm 1. The procedure takes as input the raw point cloud data in the base reference frame $P_{\text{base}} = \{\mathbf{p}_i\}_{i=1}^N$ where $\mathbf{p}_i \in \mathbb{R}^3$, the goal object pose obtained from human demonstration $\mathbf{T}_{\text{base}}^{\text{target}} \in \mathbb{R}^{4 \times 4}$, the local hole search volume defined by a cube with side $l \in \mathbb{R}^+$, the $\alpha \in \mathbb{R}^+$ parameter determining con-

cave hull computation, the point cloud of the grasped object $P_{\text{obj}} = \{\mathbf{p}_j\}_{j=1}^M$ where $\mathbf{p}_j \in \mathbb{R}^3$, and the area ratio threshold $\rho \in \mathbb{R}^+$. First, the demonstrated target $\mathbf{T}_{\text{base}}^{\text{target}}$ is assigned as the initial hole pose estimate $\mathbf{T}_{\text{base}}^{\text{hole}}$. The TransformCloud function takes as input the point cloud data defined in the base reference frame P_{base} and projects them onto the target frame $\mathbf{T}_{\text{base}}^{\text{target}}$, which yields P_{target} . The transformed cloud P_{target} is cropped using the CropCloud function to extract a volume of interest around the goal point. If the extracted volume P_{local} is not empty, a plane model is fitted to the cloud through the SACPlaneSegmentation function, which employs a sample consensus approach to return a collection of resulting plane inliers P_{inliers} and the plane parameters C_{model} . The plane parameters C_{model} are used to construct the plane $\mathcal{S} \in \mathbb{R}^2$. The ProjectInliers function projects the input point cloud of inliers P_{inliers} onto the plane \mathcal{S} , generating a flattened set of points P_{plane} . The ConcaveHull function applies an alpha-shape based concave hull extraction algorithm to its input cloud P_{plane} , producing a set of polygons $P_{\text{polygons}} = \{P_k\}_{k=1}^K$ corresponding to closed contours in the plane. The AreaRatio function compares the area of polygon P with the projection of the object model P_{obj} onto the polygon plane, returning a scalar ratio. If the areas match within the provided threshold ρ , the centroid transform $\mathbf{T}_{\text{base}}^P$ of the current polygon P is stored as a hole pose candidate $\mathbf{T}_{\text{base}}^{\text{hole}}$. The polygon with its centroid closest to the human demonstration target $\mathbf{T}_{\text{base}}^{\text{target}}$ is selected as the final hole pose estimate $\mathbf{T}_{\text{base}}^{\text{hole}}$. To achieve this, the distance d between the translation vector of the current hole estimate $\mathbf{t}^{\text{hole}} \in \mathbb{R}^3$ and each translation vector of the examined polygon centroids $\mathbf{t}^P \in \mathbb{R}^3$ is calculated.

C. Global Planning

Human demonstrations are often sub-optimal in terms of path length, containing unnecessary motion segments. To compensate for this redundancy, the trajectory segments that are not critical for the demonstrated assembly task are refined using the RRT* path planning algorithm [14].

Let the known obstacle closed space denoted $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$. For multiple obstacles, the obstacle space is defined as $\mathcal{X}_{\text{obs}} := \bigcup_{q=1}^Q \mathcal{X}_{\text{obs},q}$, where $Q \in \mathbb{N}$ is the total number of obstacles. The free open space results in $\mathcal{X}_{\text{free}} := (\mathcal{X}_{\text{obs}})^c = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$. Let the known initial start state represented as $\mathbf{x}_{\text{start}}^{\text{init}}$ and the known initial goal state as $\mathbf{x}_{\text{goal}}^{\text{init}}$. The RRT* algorithm rewires the tree by considering the potential cost-to-go to every node $v \in V$ (we refer to nodes v and states \mathbf{x} interchangeably), that lies in a circular set near the node of interest. The computational complexity of the RRT* is $\Theta(|V| \log |V|)$, where $|V|$ denotes the cardinality of the set of nodes V and $\Theta(\cdot)$ a tight bound. Thus, RRT* is a computationally demanding algorithm for large spaces. However, if we consider a small subspace, this methodology allows for real-time implementation.

The RRT* is used to compute offline the global graph $\mathcal{G}(V, E)$ which contains the global path $\pi(\mathbf{x}_{\text{start}}^{\text{init}}, \mathbf{x}_{\text{goal}}^{\text{init}})$. The RRT* is presented in Algorithm 2 and is described as follows. The Sample provides a random state \mathbf{x}_{rand} by randomly sampling the free state space $\mathcal{X}_{\text{free}}$ with a uniform distribution.

Algorithm 2: RRT* ($\mathcal{X}, \mathcal{X}_{\text{obs}}, \mathbf{x}_{\text{start}}^{\text{init}}, \mathbf{x}_{\text{goal}}^{\text{init}}, N$)

```
1  $V \leftarrow \mathbf{x}_{\text{start}}^{\text{init}}, \mathbf{x}_{\text{goal}}^{\text{init}}; E \leftarrow \emptyset;$ 
2 for  $n = 1$  to  $N$  do
3    $\mathbf{x}_{\text{rand}} \leftarrow \text{Sample}(\mathcal{X}_{\text{free}}, N);$ 
4    $\mathbf{x}_{\text{nearest}} \leftarrow \text{Nearest}(V, \mathbf{x}_{\text{rand}});$ 
5    $\mathbf{x}_{\text{new}} \leftarrow \text{Steer}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{rand}});$ 
6   if  $\text{NoCollision}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}}, \mathcal{X}_{\text{obs}})$  then
7      $X_{\text{near}} \leftarrow \text{Near}(V, \mathbf{x}_{\text{new}});$ 
8      $\eta \leftarrow \text{Line}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}});$ 
9      $(\mathbf{x}_{\text{min}}, c_{\text{min}}) \leftarrow \text{Parent}(\mathbf{x}_{\text{new}}, X_{\text{near}}, \eta);$ 
10     $V \leftarrow V \cup \{\mathbf{x}_{\text{new}}\};$ 
11     $E \leftarrow E \cup \{(\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{new}})\};$ 
12     $\mathcal{G} \leftarrow \text{Rewire}(\mathcal{G}, X_{\text{near}}, \mathbf{x}_{\text{new}});$ 
13 return  $\mathcal{G}(V, E);$ 
```

Then, the randomly sampled state \mathbf{x}_{rand} and the rest states in the set of nodes V are compared to yield the nearest state $\mathbf{x}_{\text{nearest}}$ to the random state. The function `Steer` indicates a new state \mathbf{x}_{new} which is closer to nearest state, by connecting the \mathbf{x}_{rand} and $\mathbf{x}_{\text{nearest}}$ with a steering function. In our case, the steering function follows a straight path. Next, the `NoCollision` function examines potential collisions from \mathbf{x}_{new} to $\mathbf{x}_{\text{nearest}}$ with the obstacle space \mathcal{X}_{obs} . If the path is collision free, the algorithm proceeds to the rewiring process. The function `Near` collects the set of states that lie in the circle $\|\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{nearest}}\| \leq \gamma(\log N_s / N_s)^{1/(n+1)}$, where $\gamma \in \mathbb{R}^+$ is the connection radius for the rewiring process, N_s is the number of samples, and n is the state space dimension [27]. The function `Line` connects the nearest state $\mathbf{x}_{\text{nearest}}$ to the new state \mathbf{x}_{new} with a straight line. The subroutine `Parent` identifies a parent state \mathbf{x}_{min} and a corresponding cost-to-go c_{min} from any state in the near set X_{near} to the new state \mathbf{x}_{new} . Next, the new state \mathbf{x}_{new} is added to the set of nodes V and the new edge connecting the parent state \mathbf{x}_{min} with the new state \mathbf{x}_{new} to the set of edges E . The `Rewire` function indicates the path with minimum cost, by adding or discarding edges between states in the near set X_{near} accordingly. After evaluating N_s number of samples, the process terminates and returns the graph $\mathcal{G}(V, E)$.

D. Local Replanning

The final phase of the task consists of placing the peg exactly above the hole. That is a challenging task, as the exact location of the hole is not known *a priori*. To this end, we exploit the perception system throughout the task execution to estimate the exact location of the hole. Then, we set the location of the hole as the final goal. Next, the local replanning framework is employed to modify the tail of the path accordingly to the final goal.

The overall framework is illustrated in Figure 5. The robot starts at the initial start state $\mathbf{x}_{\text{start}}^{\text{init}}$ and computes the global path $\pi(\mathbf{x}_{\text{start}}^{\text{init}}, \mathbf{x}_{\text{goal}}^{\text{init}})$. The initial goal state $\mathbf{x}_{\text{goal}}^{\text{init}}$ is selected from one of the demonstration goal states. Through the task execution, the perception system seeks to observe the final goal state of

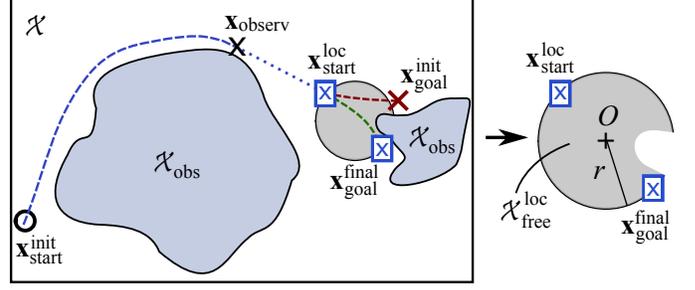


Fig. 5. The space of the global path is compressed to a sufficiently small subspace that allows the implementation of the RRT* in real time. The computed subspace on the right side of the figure is the relative complement of the obstacle space in the circular space. The geometric center O of the circular space is the center of the line, connecting $\mathbf{x}_{\text{start}}^{\text{loc}}$ and $\mathbf{x}_{\text{goal}}^{\text{final}}$. Additionally, $\mathbf{x}_{\text{observ}}$ denotes the first state at which the final goal is observed. The dotted line denotes the part of the global trajectory that the system keeps following until the replanned trajectory is executed.

the hole $\mathbf{x}_{\text{goal}}^{\text{final}}$ in real time. We name the first state at which the final goal state is observed as $\mathbf{x}_{\text{observ}}$. The corresponding traversed path from the initial start state to the observation state is shown with a blue dashed line in Fig. 5. Since the RRT* is computationally demanding, we provide sufficient time for the replanning computation, by assigning as the local replanning start state $\mathbf{x}_{\text{start}}^{\text{loc}}$ a future state after the observation of the hole. The traversed path from $\mathbf{x}_{\text{observ}}$ up to the $\mathbf{x}_{\text{start}}^{\text{loc}}$ follows the global path and is indicated with blue dotted line in Fig. 5. We condense the original space by defining a circular subspace $\mathcal{X}_{\text{circle}}^{\text{loc}} := \{\mathbf{x} \in \mathcal{X}_{\text{free}} \mid \|\mathbf{x} - O\|^2 \leq r^2\}$, based on the local replanning start state and the final goal state $O = (\mathbf{x}_{\text{start}}^{\text{loc}} + \mathbf{x}_{\text{goal}}^{\text{final}})/2$ and with radius $r = \|\mathbf{x}_{\text{start}}^{\text{loc}} - \mathbf{x}_{\text{goal}}^{\text{final}}\|$, i.e. the geometric center of $\mathbf{x}_{\text{start}}^{\text{loc}}$ and $\mathbf{x}_{\text{goal}}^{\text{final}}$. Next, the local free subspace $\mathcal{X}_{\text{free}}^{\text{loc}}$ is produced as the relative complement of the obstacle space in the circular subspace $\mathcal{X}_{\text{free}}^{\text{loc}} = \mathcal{X}_{\text{circle}}^{\text{loc}} \setminus \mathcal{X}_{\text{obs}}$, as shown in the right part of Fig. 5. The completeness of the RRT* in the local free subspace is assessed with topological connectedness tools [17]-(Lemma 3). The resulted local free subspace is sufficiently small to allow for the computation of the local path in real time. Lastly, provided the local start state $\mathbf{x}_{\text{start}}^{\text{loc}}$ and the final goal state $\mathbf{x}_{\text{goal}}^{\text{final}}$ we execute the RRT* algorithm in the $\mathcal{X}_{\text{free}}^{\text{loc}}$ to obtain a new path $\pi(\mathbf{x}_{\text{start}}^{\text{loc}}, \mathbf{x}_{\text{goal}}^{\text{final}})$.

To verify that the peg-in-hole task is completed, the system exploits the adaptive nature of the utilized gripper. After arriving to the final goal state, the end-effector executes a "Z" motion in the estimated hole plane, while tracking the object pose with respect to the camera frame. This final alignment motion can also compensate for minor hole estimation inaccuracies. When the object is detected within the hole edges, the task execution is considered successful.

V. RESULTS AND DISCUSSION

To validate the implemented system experimentally, a group of five able-bodied subjects (male, ages: 26 ± 1) was instructed to demonstrate the peg-in-hole task using three different approaches: pendant-based teaching, kinesthetic teaching, and dataglove-based teleoperation. The pendant-based method

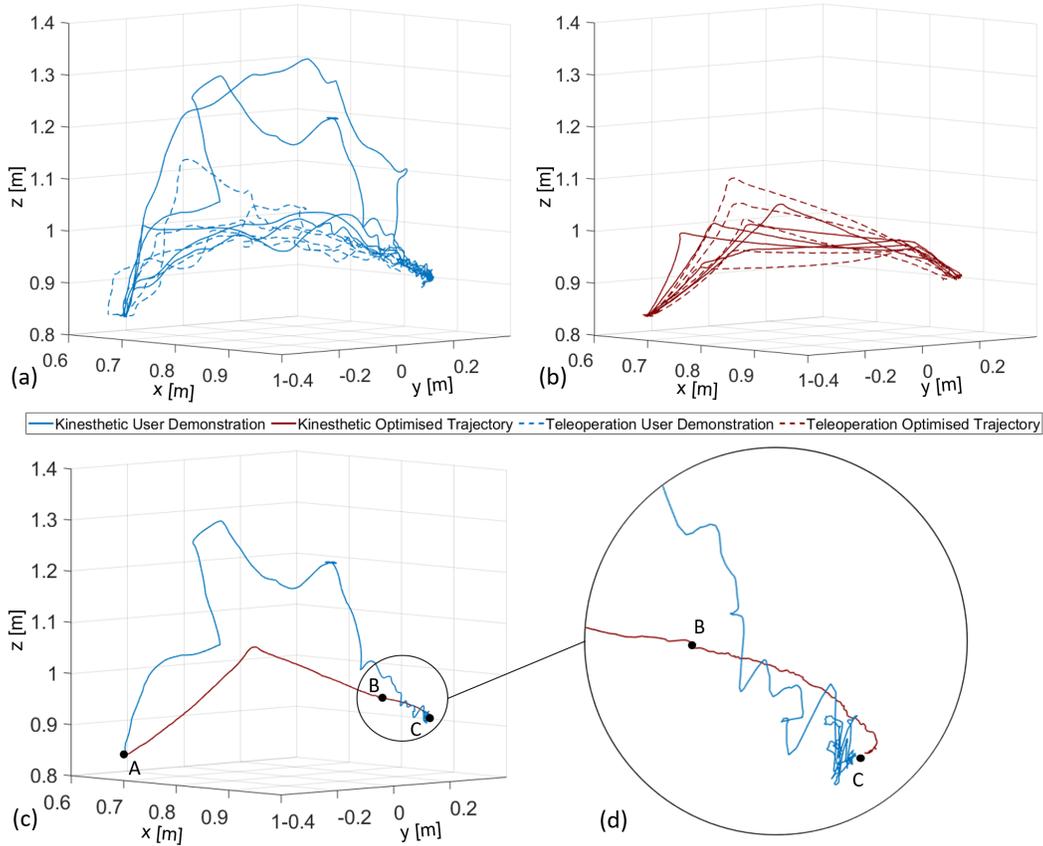


Fig. 6. Comparison of object trajectories for (a) kinesthetic and teleoperated user demonstrations, and (b) optimized trajectory executions of the corresponding demonstrations. Subfigures (c) and (d) isolate a demonstrated and optimized trajectory obtained through kinesthetic teaching. More precisely, subfigure (c) presents the full motion from grasp to sample placement, and subfigure (d) focuses on the local replanning. In subfigure (c), point A corresponds to the object grasping pose, point B to the initial pose for local replanning, and point C represents the sample insertion / placement point. The sharp rerouting of the optimized trajectory is due to the geometry and structure of the obstacle space.

served as a comparison baseline, utilizing the translation and rotation jog interface on the robot control box to guide the end-effector to the desired position. The demonstrations obtained through teleoperation and kinesthetic teaching were examined with- and without path optimization and local replanning enabled. For each experiment, the sample was placed in a pre-defined initial position on a tray adjacent to the experimental platform. Only demonstrations that resulted in the successful execution of the assembly task have been considered for comparison and optimization. After completing the teaching experiments, subjects were instructed to complete a survey asking them to evaluate the three demonstration approaches.

The first stage of the system evaluation consisted of a comparison between the different demonstration approaches in terms of teaching time and ease of use. For each subject and demonstration approach, the teaching time was measured as the time between grasping the object on the tray and releasing it after successful insertion. The ease of use was obtained through the survey, where the subjects were asked to evaluate teaching difficulty on a scale of 0 to 10 (0 being very easy and 10 being very difficult). The obtained average teaching time and ease of use values for each demonstration approach are collected in Table I. The kinesthetic teaching approach

has proven to be the fastest and most user friendly among the three, with subjects completing the task nearly four times faster than with the pendant. Even though teleoperation was reported to be very engaging, it was rather difficult to use due to the necessary safety distance from the operated robot. It must be noted, however, that the robot platform was not as large compared to the human user, which contributed to the better kinesthetic teaching score. For bigger robots, the teleoperation approach would likely be more appropriate from a safety and accessibility perspective.

The demonstrated and optimized object trajectories obtained through kinesthetic teaching and teleoperation are presented in Fig. 6, where a single pair of kinesthetic teaching trajectories is isolated in subfigures (c) and (d) to allow for closer inspection. Examining the figures, it is clear that the raw user paths are longer and contain jittery motion that compromises execution efficiency. Such perturbations are particularly evident closer to the goal, where the user needs to guide the robot end-effector (adaptive gripper) with greater precision. Directly replaying a demonstration results in speed fluctuations and rarely results in successful sample insertion due to grasping and positioning inaccuracies. The teleoperated demonstration path lengths were on average 20% longer than the kinesthetic

TABLE I
AVERAGE TEACHING TIME, EASE-OF-USE SCORE, AND TRAJECTORY LENGTH COMPARISON

| Teaching Method | Teaching Time [s] | Ease of Use [0–10] | User Trajectory [m] | Optimized Trajectory [m] |
|-----------------|-------------------|--------------------|---------------------|--------------------------|
| Pendant | 85.16 | 5.80 | / | / |
| Kinesthetic | 22.15 | 1.00 | 1.02 | 0.81 |
| Teleoperation | 49.75 | 5.00 | 1.21 | 0.86 |

demonstrations, as the users need significantly more time and alignment effort in teleoperation due to the safety distance and decreased visibility. The optimized trajectories, on the other hand, are much smoother. The local replanning (from point B to point C, depicted in Fig. 6) took on average less than 0.1 s due to the constrained search space and introduced no delay in the overall task execution. The optimized kinesthetic teaching and teleoperation trajectories were on average 20.3% and 28.9% shorter than the demonstrations, respectively. Overall, the optimized trajectories were on average 26.4% shorter than the demonstrations across all teleoperation and kinesthetic teaching trials. The experiments involving both the demonstrated and the optimized trajectories are presented in the accompanying video, which is available in HD quality at the following URL: www.newdexterity.org/skilltransfer

VI. CONCLUSION

This work focused on combining programming by demonstration with path optimization and local replanning methods to facilitate a fast and intuitive execution of assembly tasks requiring minimal user expertise and involvement. The framework can also be extended to enhance task programming in service robotics. Two demonstration approaches were developed, integrated into the framework, and compared: i) kinesthetic teaching and ii) teleoperation based teaching. The two approaches have also been compared with the classic, pendant based teaching. Both developed methods require a trajectory optimization step that transforms the demonstrated robot trajectories considering the obstacle space and the provided task specifications. The local replanning adjusts the optimized robot path based on online feedback from the perception system, guaranteeing a collision-free navigation and the efficient task execution. The efficiency of the methods has been validated with a series of experiments involving the execution of a complex assembly task with a robot arm and gripper system. The different approaches have been evaluated in terms of teaching time, ease of use, and path length.

REFERENCES

- [1] 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, May 2009.
- [2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot Programming by Demonstration*. Springer Berlin Heidelberg, 2008, pp. 1371–1394.
- [3] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.
- [4] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, 2018.

- [5] K. Fischer, F. Kirstein, L. C. Jensen, N. Krüger, K. Kukliński, M. V. aus der Wieschen, and T. R. Savarimuthu, "A comparison of types of robot control for programming by demonstration," in *ACM/IEEE Intern. Conference on Human-Robot Interaction*, 2016, pp. 213–220.
- [6] A. J. Ijzpeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1398–1403.
- [7] M. V. Liarokapis, C. P. Bechlioulis, G. I. Boutselis, and K. J. Kyriakopoulos, *A Learn by Demonstration Approach for Closed-Loop, Robust, Anthropomorphic Grasp Planning*. Springer International Publishing, 2016, pp. 127–149.
- [8] M. Liarokapis, C. P. Bechlioulis, P. K. Artemiadis, and K. J. Kyriakopoulos, "Deriving Humanlike Arm Hand System Poses," *Journal of Mechanisms and Robotics*, vol. 9, no. 1, 2017.
- [9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [10] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.
- [11] K. Ota, D. K. Jha, T. Oiki, M. Miura, T. Nammoto, D. Nikovski, and T. Mariyama, "Trajectory optimization for unknown constrained systems using reinforcement learning," *arXiv:1903.05751*, 2019.
- [12] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.
- [13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [15] M. Otte and E. Frazzoli, "RRT^X: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The Inter. Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.
- [16] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.
- [17] G. P. Kontoudis and K. G. Vamvoudakis, "Kinodynamic motion planning with continuous-time Q-learning: An online, model-free, and safe navigation framework," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3803–3817, 2019.
- [18] C.-M. Chang, L. Gerez, N. Elangovan, A. Zismatos, and M. Liarokapis, "On alternative uses of structural compliance for the development of adaptive robot grippers and hands," *Frontiers in Neurobotics*, vol. 13, p. 91, 2019.
- [19] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, June 1989.
- [20] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.
- [21] S. Chitta *et al.*, "ros_control: A generic and simple control framework for ROS," *The Journal of Open Source Software*, 2017.
- [22] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9-13 2011.
- [23] I. A. Sucan and S. Chitta. MoveIt. [Online]. Available: <http://moveit.ros.org>
- [24] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.
- [25] A. G. Buch, D. Kraft, J. Kamarainen, H. G. Petersen, and N. Krüger, "Pose estimation using local structure-specific shape and appearance context," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2080–2087.
- [26] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor fusion IV: Control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.
- [27] K. Solovey, L. Janson, E. Schmerling, E. Frazzoli, and M. Pavone, "Revisiting the asymptotic optimality of RRT," *arXiv:1909.09688*, 2019.