Flexible Automation Driven by Demonstration: Leveraging Strategies that Simplify Robotics

Andrea Giusti[1,2,5], Martijn J.A. Zeestraten[3,5], Esra Icer[2], Aaron Pereira[2], Darwin G. Caldwell[3], Sylvain Calinon[4,3], Matthias Althoff[2]

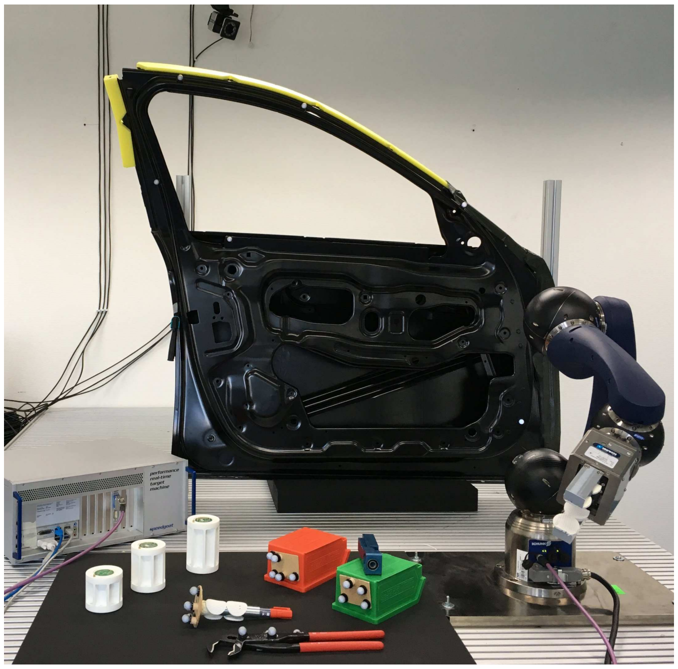


Fig. 1: Modular robot setup used in the experiment.

I. Reconfigurable and automatically programmable robots

How do I automate this task? This is a crucial question for production engineers. In classical industrial automation—think of a car factory—robots perform a small set of tasks for long periods of time. They were selected because their kinematic structure and strength suited the task requirements, and their motions were pre-programmed by a skilled programmer. In flexible manufacturing environments, tasks may change daily or hourly. The classical approach to automation is less suitable here: Buying a dedicated robot for each set of tasks is uneconomical and coding each task is too time consuming. Further, qualified programmers with the requisite knowledge might not be available. To solve these problems, we propose a framework for modular robots that determines their structure and program based on human demonstrations.

Programming by Demonstration (PbD) [1], [2] aims at programming robots solely based on demonstrations. Quick and user-friendly, PbD can replace teach pendants or text-based command interfaces, which are currently common in industry. It is a user-friendly method for skill transfer because it leverages a learning strategy natural to humans: *imitation*.

Just as PbD offers accessibility and flexibility in terms of programming, modular robots offer these benefits in terms of robot assembly and control. Although anthropomorphic manipulators seem an obvious choice to mimic human demonstrations, a single manipulator is not likely to be capable of performing the large variety of tasks found in industry. For example, while demonstrating the task, the human may need to step (in order to reach) or use several redundant degrees of freedom in their body; these DOFs may not be necessary in a robot optimized for the task. Our approach constructs a task-specific serial kinematic manipulator with the required degrees of freedom: this will have the required range of motion, but cost less than full-sized humanoids.

However, it is neither trivial to find the optimal module assembly, nor to control the resulting robot robustly and precisely. The large design space makes modular robot synthesis complex and time-consuming. To tackle this, both exhaustive search-based and sampling-based methods have been used [3]–[6]; the former give consistent results but the latter are computationally more efficient. Our framework generates the most suitable robot assembly for the demonstrated task (e.g. the most energy efficient), based on Icer *et al.* [4] using an exhaustive yet efficient hierarchical composition synthesis.

The modularity of our approach makes the control design more complex since each assembly exhibits different kinematics and dynamics. Most control approaches for modular robots are decentralized and are either model-free (limiting performance) or model-based (involving communication between modules). In this paper, however, we use a simple approach for on-the-fly generation of centralized model-based controllers [7], [8]. This approach automatically derives the kinematic and dynamic description of the robot directly after assembly by processing electronically-transmitted module data. In contrast to decentralized architectures, our approach can avail of well-established centralized control techniques, including tracking controllers and dynamic scaling of trajectories.

Our work combines methods in PbD and modular robots for the first time. This combination yields a flexible and user-friendly robotic system which automatically finds the most suitable assembly of modules and generates the controller,

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's 7th Framework Programme FP7/2007-2013/ under REA grant agreement number 608022.

[1]Fraunhofer Italia Research s.c.a.r.l, Innovation Engineering Center (IEC), 39100 Bolzano, Italy. andrea.giusti@fraunhofer.it,

[2]Department of Informatics, Technische Universität München, 85748 Garching, Germany. {andrea.giusti, esra.icer, aaron.pereira, matthias.althoff}@tum.de,

[3] Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy. {martijn.zeestraten, darwin.caldwell}@iit.it

[4] Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland. sylvain.calinon@idiap.ch

[5] These authors contributed equally to this work.

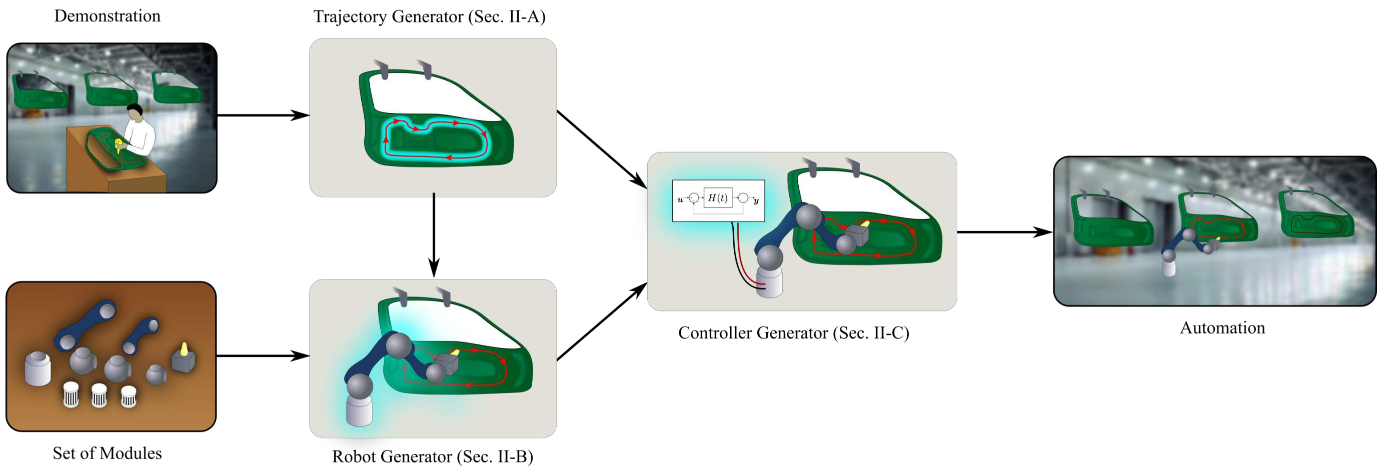

Demonstration
Trajectory Generator (Sec. II-A)
Controller Generator (Sec. II-C)
Automation
Set of Modules
Robot Generator (Sec. II-B)
u
$H(t)$
y

Fig. 2: Overview of the method: A demonstrator first demonstrates a task with a demonstrator tool. The trajectory generator converts this into a trajectory, which is used by the robot generator to find the optimal robot assembly from the available modules. Finally, the controller is generated and the task is automated.

from a task demonstrated by an operator. To program a different task, the operator simply demonstrates it, reassembles the robot in the most suitable assembly suggested, and runs the newly generated controller: reassembly and reprogramming are achieved in one go. This approach provides several (cost-saving) advantages over anthropomorphic robots for small and medium enterprises: the ability to reconfigure makes a single system applicable to a wider range of tasks (flexibility); the amount of expert knowledge required to program, assemble and control a robot is low compared to conventional approaches (ease of use); the task-specific optimization reduces the number of moving parts (reduced maintenance); modular robots are easier to maintain as their parts are designed to be replaced (less down time).

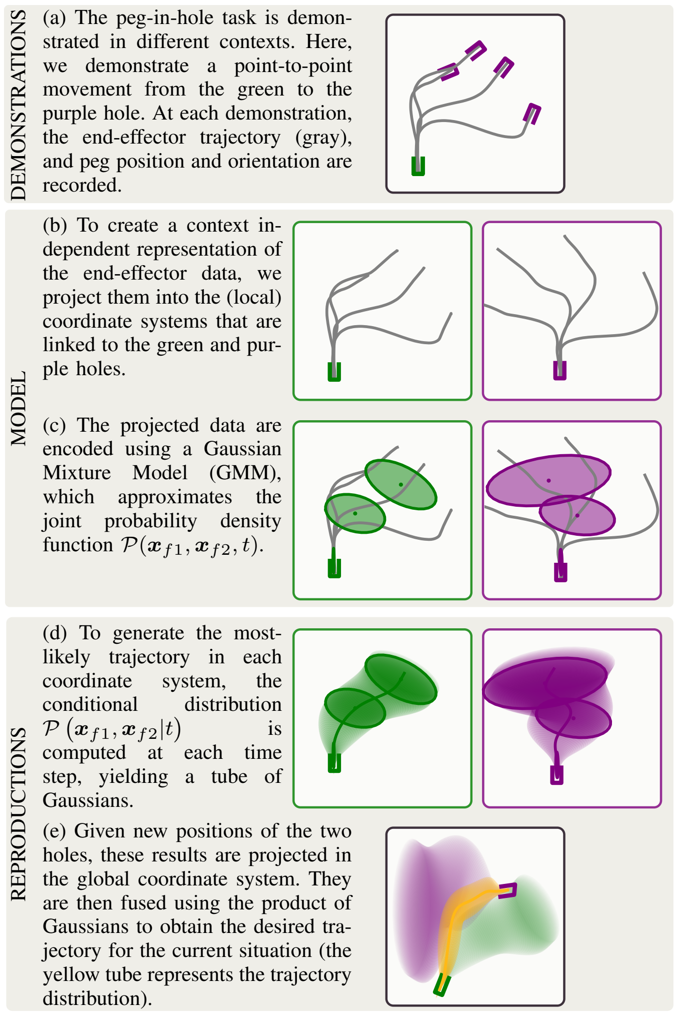
We next detail the complete approach and demonstrate it in two different applications using the modular robot setup in Fig. 1.

II. A UNIFIED APPROACH TO FLEXIBLE AUTOMATION

Figure. 2 illustrates our concept. A human operator has a task to automate, dedicated tools to demonstrate the task (see e.g. Fig. 5), and a set of robot modules. First, the user performs the task using the demonstrator tool to provide the system with demonstration data. Next, she places all task-relevant objects into their intended positions for production. The object locations are sensed and delivered together with the demonstration data to the *Trajectory Generator*. As detailed in Section II-A, the data are encoded in a task model from which the desired task-space trajectory is generated. This trajectory is fed to the *Robot Generator* which finds the most suitable module assembly as described in Section II-B. Finally, after assembly, the low-level motion control (*Controller Generator*) executes the demonstrated task as detailed in Section II-C.

A. User-Friendly Task Transfer

Demonstration data are core to our approach. They can be obtained in various ways: Users can record the robot state



(a) The peg-in-hole task is demonstrated in different contexts. Here, we demonstrate a point-to-point movement from the green to the purple hole. At each demonstration, the end-effector trajectory (gray), and peg position and orientation are recorded.

(b) To create a context independent representation of the end-effector data, we project them into the (local) coordinate systems that are linked to the green and purple holes.

(c) The projected data are encoded using a Gaussian Mixture Model (GMM), which approximates the joint probability density function $\mathcal{P}(\boldsymbol{x}_{f1}, \boldsymbol{x}_{f2}, t)$.

(d) To generate the most-likely trajectory in each coordinate system, the conditional distribution $\mathcal{P}\left(\boldsymbol{x}_{f1}, \boldsymbol{x}_{f2} | t\right)$ is computed at each time step, yielding a tube of Gaussians.

(e) Given new positions of the two holes, these results are projected in the global coordinate system. They are then fused using the product of Gaussians to obtain the desired trajectory for the current situation (the yellow tube represents the trajectory distribution).

Fig. 3: Illustration of the task-parameterized model.

while manually controlling the robot through teleoperation, or by physically interacting with it (kinesthetic teaching). Alternatively, they can record themselves performing the task and transfer these data to the robot. Since the robot is not assembled at the time of demonstration (as its structure is to be determined) we choose the latter method.

To let PbD surpass simple record-and-replay behavior, the robot must understand the objective underlying the demonstration data; it must comprehend *what to imitate*, as such insight lets it generalize skills to new situations. For example, while it may not understand “slice” a “pancake” in “half”, it understands that it should move its end-effector above a certain object, align the end-effector plane perpendicular to the plane of this object, and move it downwards until a certain point on the object. Given this, the robot must use its body to achieve imitation. This requires the robot to know *how to imitate*. Any method for PbD must address these core questions, which typically involves pre-structuring the task models.

1) What to imitate: We use a Riemannian Task-Parameterized Gaussian Mixture Model (TP-GMM) to model the task [9]. This model can adapt a demonstrated task to new contexts, and allows the robot to handle both position and orientation data. Context adaptation is achieved by representing the task in different (local) coordinate systems, as illustrated in Fig. 3. These coordinate systems are linked to task-relevant objects or landmarks (the task context), and related to a global coordinate system.

By considering the demonstration data from different perspectives, the context-related robot motions and their importance in task execution can be determined. When demonstrations are given in different task contexts (Fig. 3(a)), variant and invariant regions appear in the local coordinate systems (Fig. 3(b, c)). The observed variance relates to the importance of an object or landmark during task execution: invariance indicates that the end-effector moved consistently with respect to an object, and variance indicates that the object state was irrelevant to the end-effector motion. These local motions and their (co)variance are captured in the TP-GMM: this model represents the robot’s understanding of the task, i.e. *what to imitate*.

2) How to imitate: Having modelled the task, we obtain a trajectory—*how to imitate*—as follows. We detect objects and their coordinate systems in task space, and transform the TP-GMM into task-space coordinates (Fig. 3(d)). Using the product of Gaussians, we obtain a distribution of trajectories in the task space, whose mean is the desired trajectory (Fig. 3(e)).

Furthermore, the robot must know how to relate the demonstrated data to its kinematic structure: a mapping is required to relate the demonstrator state to the robot state. This classical problem is called the *correspondence problem* [2]. We resolve it practically, by working in task space using dedicated demonstrator tools (see Section. III).

3) Encoding Orientation: The use of task space requires us to cope with the non-Euclidean space of orientation data. Unlike position data, statistical operations on orientation data are not straightforward as they do not allow Euclidean operations (e.g. the sum of two rotation matrices is not a rotation matrix). Riemannian geometry and statistics provide a way to perform PbD on orientation data [9].

Riemannian TP-GMM has several of advantages over other well-known models such as Dynamic Movement Primitives (DMP) [10] and Probabilistic Movement Primitives (ProMP) [11]. Whereas DMP only adapts to changes in goal position, TP-GMM encodes relations to landmarks anywhere along the motion trajectory. Furthermore, as TP-GMM encodes spatial relations, it also extracts soft motion constraints from the demonstration data. Fig. 3 shows how this allows the motion to adapt to different orientations of start and goal landmarks. Like TP-GMM, a ProMP allows adaptation of the trajectory while respecting movement constraints. However, its structure is inherently Euclidean, and therefore not well-suited for orientation data.

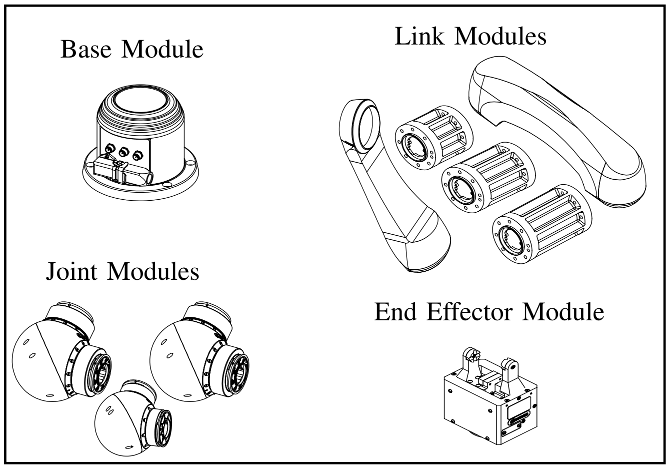
4) Context-dependent movement adaptation: Movement adaptation in DMP or ProMP requires the specification of a new end-effector (goal) position explicitly. In the peg-in-hole example, one must specify the exact center of the hole to allow the DMP or ProMP to reach it. In contrast, TP-GMM only requires location and orientation of a coordinate system linked to the hole. These coordinate systems adapt not only the end-point, but also the direction of the movement for the approaching phase. They can be arbitrarily set without prior knowledge of the task at hand, making TP-GMM more generic than DMP or ProMP. For the peg-in-hole example, the origin of the coordinate system linked to the hole does not need to lie at its center; the required relation between end-effector and hole is inferred from demonstration data.

B. Robot Assembly

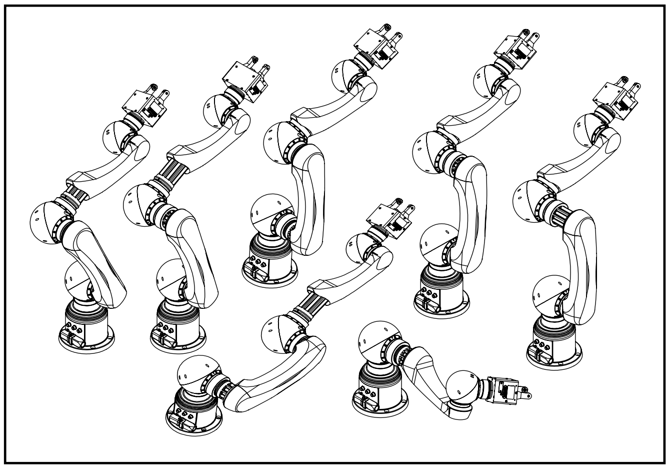
The set of modules in our setup comprises a base, three joint modules, five link modules and one end-effector (see Fig. 4a). The number of possible assemblies from a set of modules grows rapidly with the size of the set, making selection of the optimal assembly for a task challenging. To solve this combinatorial problem, we eliminate assemblies hierarchically [4]. We perform a series of increasingly computationally expensive tests on all assemblies and eliminate unfeasible ones at each step, as early as possible. From the remaining feasible assemblies, we pick the best (in our case, that which can achieve the task fastest).

Our first test eliminates all assemblies which do not have the required Degrees of Freedom (DOFs). Next, we sample end-effector poses from the trajectory given by the Trajectory Generator in Section II-A, and first eliminate assemblies whose total length is less than the distance from the base to these poses, then assemblies that cannot achieve the poses kinematically and statically are removed, and finally those assemblies that generate self-collision are discounted. For inverse kinematics we use an iterative solver initialized from several points to improve the chances of finding a valid solution. Self-collision is detected as in [4]: enclosing module geometry in spheres or cylinders, and determining collision from pairwise distance checks.

From the remaining set of feasible assemblies, we generate the trajectory in joint space using the inverse kinematics scheme of Section II-C and scale speed such that joint positions and torque limits are respected. The optimal assembly is that which can perform the task fastest. Other optimality criteria may be considered (e.g. minimal energy consumption).



(a)



(b)

Fig. 4: (a) The given modules and (b) a set of possible assemblies. The sketches of the robot components have been derived using CAD data which has been downloaded from the website of the robot manufacturer (Schunk GmbH).

If no robot is found at the assembly selection stage, the path is considered unfeasible for the available set of modules. The time required for obtaining this result depends on the number of modules available, their geometry, and the actual path.

C. Robot Control

Once the assembly is selected and the user has assembled the robot, the controller should make the arm perform the task without additional user intervention, which would otherwise limit the swift reconfigurability of the overall system. The realization of such a controller is nontrivial, particularly since the kinematics and dynamics of the assembled robot depend on the modules and their configuration.

To solve this challenge, we automatically generate centralized model-based controllers from kinematic and dynamic parameters of the modules [7], [8]. Each module has a unique ID and a set of characterizing parameters (module data), stored either in the module or in a database. The module data are based on an extension of the Denavit-Hartenberg (D-H) convention as in [7], which is necessary to resolve the typical non-uniqueness of the standard D-H notation for certain relative joint axis orientations. After assembly, the module data are collected by the central control unit which generates a kinematic and dynamic model of the robot, from which the controller is derived directly [8].

We denote the array of structures created by the central control unit by $\mathbf{ModRob}$. It contains the module data and order of assembly. For an assembled robot with N degrees of freedom, we wish to track sufficiently smooth joint-space trajectories $\mathbf{q}_d \in \mathbb{R}^N$ and automatically deploy a classical passivity-based tracking controller. We denote by $NE^*_{mod}(\cdots, \mathbf{ModRob})$ the algorithm that synthesizes $\mathbf{ModRob}$ into a description of the assembled robot as in [8] and performs the modified recursive Newton-Euler (N-E) method for passivity-based control as in [12]. By denoting respectively with $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$, $\mathbf{n}(\mathbf{q},\dot{\mathbf{q}})$ the inertia matrix, the vector of Coriolis and centrifugal terms and the vector of friction and gravity terms, this algorithm efficiently provides

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}_a + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = NE^*_{mod}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}},\mathbf{ModRob}),$$

where $\mathbf{q} \in \mathbb{R}^N$ is the vector of joint positions. Then, the passivity-based control commands are computed as:

$$\mathbf{u} = NE^*_{mod}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{ModRob}) + \mathbf{K}\,\mathbf{r},$$

where

$$\dot{\mathbf{q}}_a = \dot{\mathbf{q}}_d + \mathbf{K}_V(\mathbf{q}_d - \mathbf{q}), \quad \mathbf{r} = (\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_V(\mathbf{q}_d - \mathbf{q}),$$

and where $\mathbf{K}$ and $\mathbf{K}_V$ are positive definite matrices of proper dimension. Note: for each new assembly, the structure $\mathbf{ModRob}$ changes accordingly, so that the low-level control is always computed with the correct model and thus always ensures global tracking.

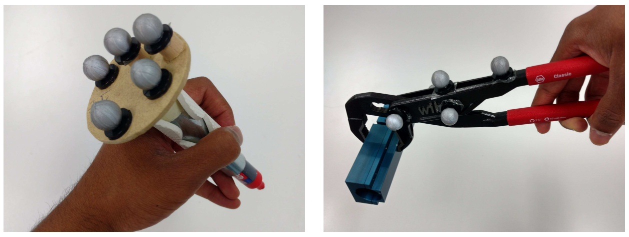
To track the encoded trajectories in task-space and provide the proper reference $\mathbf{q}_d$ to the passivity-based tracker mentioned above, we solving the inverse kinematic problem numerically online [13]. By denoting with $\mathbf{p}_r \in \mathbb{R}^3$ the desired end-effector trajectory for the translation, with $\mathbf{o}_r = [\eta_r, \boldsymbol{\epsilon}_r]^T \in \mathbb{R}^4$ the orientation unit quaternion, and with $\mathbf{J}^\dagger$ the Jacobian pseudo-inverse (or damped least-squares inverse [14] near the kinematic singularities), the following kinematic control scheme is employed[1]:

$$\ddot{\mathbf{q}}_d = \mathbf{J}^\dagger(\mathbf{q}_d)\Big(\boldsymbol{\nu} - \dot{\mathbf{J}}(\mathbf{q}_d,\dot{\mathbf{q}}_d)\mathbf{q}_d\Big) - \kappa\Big(\mathbf{I} - \mathbf{J}^\dagger(\mathbf{q}_d)\mathbf{J}(\mathbf{q}_d)\Big)\dot{\mathbf{q}}_d, \tag{1}$$

with

$$\boldsymbol{\nu} = \begin{bmatrix} \ddot{\mathbf{p}}_r + K_v(\dot{\mathbf{p}}_r - \mathbf{J}_p(\mathbf{q}_d)\dot{\mathbf{q}}_d) + K_p(\mathbf{p}_r - \mathbf{p}_{fk}(\mathbf{q}_d)) \\ \dot{\boldsymbol{\omega}}_r + K_\omega(\boldsymbol{\omega}_r - \mathbf{J}_\omega(\mathbf{q}_d)\dot{\mathbf{q}}_d) + K_o\mathbf{e}_o(\mathbf{q}_d) \end{bmatrix}. \tag{2}$$

[1]When moving the robot from its rest position to the initial pose of the trajectory, we wait until the inverse kinematic solution converges before moving with a point-to-point motion in joint-space.



(a) Spring loaded pen (b) Pickup Tool

Fig. 5: Demonstrator tools used in the experimental evaluation.

In (1) and (2), κ, K_v, K_p, K_ω, K_o are positive gains, $\mathbf{p}_{fk}(\mathbf{q}_d)$ is the position of the end effector computed with the forward kinematics and finally

$$\mathbf{e}_o = \eta_{fk}(\mathbf{q}_d)\boldsymbol{\epsilon}_r - \eta_r\boldsymbol{\epsilon}_{fk}(\mathbf{q}_d) - \mathbf{S}(\boldsymbol{\epsilon}_r)\boldsymbol{\epsilon}_{fk}(\mathbf{q}_d)$$

is the quaternion-based orientation error feedback vector in which $\mathbf{S}(\cdot)$ is the common skew symmetric matrix operator; $\eta_{fk}(\mathbf{q}_d)$ and $\boldsymbol{\epsilon}_{fk}(\mathbf{q}_d)$ are the components of the unit quaternion for orientation computed with the forward kinematics. Since the assembled robot may be kinematically redundant, we add damping in the null space to prevent floating null-space motions as in [15]. We emphasize that, given the automatically generated D-H table from ModRob, no user intervention is required after reassembling the robot.

III. Application

Two common types of automation tasks are *trajectory tracking*, where a robot end effector must follow a particular trajectory, and *pick and place tasks* (P&P), where a robot must grasp an object and deposit it at another location. Welding and gluing are examples of trajectory tracking, while bin picking is an example of P&P. In this section, we demonstrate our approach on both types of task.

A. Experimental Setup

The skill transfer uses a Vicon infrared motion tracking system. Though unconventional in industry, the system is easy to set up and provides accurate tracking of any object equipped with markers. The different tasks are demonstrated with *demonstrator tools* (see Fig. 5), each relating to a specific end-effector module. The known kinematic relation between the demonstrator tool and the corresponding end-effector module enables accurate transfer of end-effector pose trajectories.

During demonstration, we record poses of both the demonstrator tool and of objects that are relevant to the task (e.g. door pose, P&P locations). Irrelevant data at the start and end of each demonstration are manually cut using a graphical user interface, and the temporal signals are rescaled linearly to the interval $[0, 1]$. For reproduction, we optimize the velocity profile along the trajectory to improve performance and respect joint limits. Although the optimization might alter the demonstrated motion dynamics, in many industrial applications, including our examples, successful task execution does not depend on this.

Our modular robot comprises the Schunk LWA-4P arm and additional modules we manufactured to enhance its reconfigurability (see Fig. 4). The low-level control is implemented using Simulink Real-Time and a Speedgoat Real-Time target machine equipped with a CAN-bus communication module. The sampling rate used for control is $500\,Hz$.

B. Experimental Results

Our industrial partner BMW provided a typical scenario they would automate on their shop floors: fixing insulation material onto a car door. This skill requires pressing a previously mounted piece of fabric along a specific trajectory. An overview of the experimental setup and the results obtained are provided in Fig. 6. The skill is transferred using 3 demonstrations. To attain the required pressure, we used a spring-loaded pen tool (Fig. 5a). By pressing the tool firmly against the door frame while tracking the desired trajectory, the required pressing force can be achieved. Following the procedures laid out in sections II-A and II-B, we obtain the skill model from the demonstration data, and the module assembly. The task model in Fig. 6 displays the demonstration data (in red), the resulting skill model (green colored ellipses), and the reproduction (in blue). As the tracking task requires large precision, the Gaussians display low variance perpendicular to its trajectory. The generated trajectory mimics all essential motion features for proper profile tracking.

The procedure for finding the optimal robot assembly starts with the search for all assemblies that could fulfill the task. With the given modules, more than 9.8 million module combinations can be generated. Based on the assumptions that: *i*) all assemblies start with a base-module and end with an end-effector module, and, *ii*) two joint modules cannot be assembled consecutively; only 83 assemblies remain. From these, we eliminated assemblies that could not perform the task kinematically or statically (70), or encountered collision (9). From the remaining four assemblies we selected the one that could perform the task fastest without exceeding the joint limits. The optimal robot assembly is displayed in the center of Fig. 6.

After the robot is assembled, the controller is automatically generated as described in Section II-C. Finally, the task is replicated using the learned skill model and the robot controller. Snapshots of a reproduction are displayed at the bottom of Fig. 6.

To demonstrate the flexibility of the approach, we use the same experimental setup to transfer a P&P task, as shown in Fig. 7. As in the door task, a dedicated demonstrator tool (Fig. 5b) is used to capture the data. The task is to transport an object from the red *pick* location to the green *place* location (see e.g. top of Fig. 7). Additionally, we wish the robot to replicate the learned skill for previously unseen combinations of pick and place locations. To achieve this, we demonstrate the task on 9 different combinations of pick and place locations. The demonstrations allow the robot to determine the importance of the pick and place locations in different sections of the trajectory. This is reflected in the Gaussian Mixture Model (GMM) fitting the data set, as shown in Fig. 7, by the colored ellipsoids. The algorithm clearly distinguishes the variant and invariant regions required to reproduce the task; low variant regions appear in the pick and place frames at pick and place actions, respectively.

The optimal assemblies of the robot are determined based on the demonstration data, and on the desired pick and place locations. The approach for selecting the optimal assembly follows the same steps of the previous task. In this case, the total time required for finding the optimal assembly was under 5 minutes. At the bottom of Fig. 7 are two successful replications of the pick and place task in previously unseen situations. As the task configuration changes, the ability of the robot assembly to execute the task needs to be assessed. This is achieved by applying the inverse kinematics solver in advance.

IV. Discussion

Our proposed approach has three fundamental steps: trajectory generation, assembly selection and automatic controller generation. Here, we discuss open challenges and possible improvements for each step.

The number of demonstrations required very much depends on the task to be transferred (namely, if there are possible variations), and on the teaching propensity of the demonstrator (namely, if the user shows useful variations of the same task to the robot learner). Then, the amount and quality of the demonstrations will influence the generalization capability of the approach, meaning that the number of demonstrations also depends on the generalization capability that is expected during reproduction. Furthermore, obstacles have not been considered in the trajectory generation. This is an interesting research problem, whose solution would enhance this framework significantly. A potential approach for modulating the trajectory around obstacles is presented in [16].

The task encoding requires the selection of the number of Gaussians. Too few Gaussians, and fine movements required for successful task execution will be omitted; too many will overfit the data, and introduce movement artefacts that were not demonstrated. Although methods exist to automatically select the number of Gaussians, they are not ideal for trajectory data.

When a robot is expected to change tasks frequently, reconfiguration of the robot might not be the most time efficient alternative. This aspect will be considered in future work by enhancing the optimization procedure with the inclusion of reconfiguration cost, and by optimizing over a set of tasks rather than just one. Yet, to (optimally) execute dissimilar tasks different robots are required. Our modular approach allows such task-specific optimization, while in the past this would have required the acquisition of dedicated robots.

The presented approach assumes that the kinematic parameters of the modules and the location of the base module are accurately known, and that the tracking system is calibrated. However, the reader should keep in mind that high-precision positioning (say better than a human) requires the approach to be extended, e.g., using a scale model for demonstrations and spatially scaling human motions.

Certainly, an interesting extension of the motion controller would be the inclusion of automatic generation of an impedance control scheme starting from the automatically generated robot description from [7], [8], which could potentially be combined with the demonstration-driven encoding of the impedance target parameters as in [17], [18]. In addition, approaches to optimize the redundancy resolution, instead of simply adding damping for the null-space motions, could be also considered.

V. Conclusion

Motivated by the increasing need for easy implementation of automation in small and medium-sized enterprises, we present a scheme whereby a task is demonstrated by a human, and the optimal modular robot assembly as well as the control for this task is automatically generated. Despite tremendous progress in PbD in recent years, the ability to reproduce demonstrated skills may be limited by the structural capability of classical fixed-structure robots. With the proposed framework, we resolve such hardware limitations by making the physical structure fully adjustable: we assemble a robot from modules, which are selected to optimally fit the task programmed by demonstration. We also remove limitations to modular robot technology by performing trajectory planning using PbD and by generating robust motion control automatically.

Our framework, for the first time, combines recent developments in PbD, assembly selection, and controller synthesis of modular robots. It enables the exploitation of modular robot reconfigurability and paves the way for flexible automation.

References

[1] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," in *Handbook of Robotics 2nd Edition*, B. Siciliano and O. Khatib, Eds. Secaucus, NJ, USA: Springer, 2016, pp. 1995–2014.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[3] C. J. Paredis and P. K. Khosla, "Kinematic design of serial link manipulators from task specifications," *The International Journal of Robotics Research*, vol. 12, no. 3, pp. 274–287, June 1993.

[4] E. Icer, A. Giusti, and M. Althoff, "A task-driven algorithm for configuration synthesis of modular robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2016, pp. 5203–5209.

[5] E. Icer and M. Althoff, "Cost-optimal composition synthesis for modular robots," in *Proc. of the IEEE Multi-Conference on Control Applications (CCA)*, September 2016, pp. 1408–1413.

[6] E. Icer, H. A. Hassan, K. El-Ayat, and M. Althoff, "Evolutionary cost-optimal composition synthesis of modular robots considering a given task," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, September 2017, pp. 1408–1413.

[7] A. Giusti and M. Althoff, "Automatic centralized controller design for modular and reconfigurable robot manipulators," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, September 2015, pp. 3268–3275.

[8] ——, "On-the-fly control design of modular robot manipulators," *IEEE Trans. on Control Systems Technology*, vol. –, no. –, pp. –, – 2017 (in press).

[9] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 3, pp. 1240–1247, June 2017.

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

[11] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” in *Advances in Neural Information Processing Systems 26*, December 2013, pp. 2616–2624.

[12] A. De Luca and L. Ferrajoli, “A modified Newton-Euler method for dynamic computations in robot fault detection and control,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, May 2009, pp. 3359–3364.

[13] S. Chiaverini and B. Siciliano, “The unit quaternion: A useful tool for inverse kinematics of robot manipulators,” *Systems Analisis, Modelling and Simulation*, vol. 35, no. 1, pp. 45–60, September 1999.

[14] F. Caccavale, S. Chiaverini, and B. Siciliano, “Second-order kinematic control of robot manipulators with jacobian damped least-squares inverse: theory and experiments,” *IEEE/ASME Trans. on Mechatronics*, vol. 2, no. 3, pp. 188–194, September 1997.

[15] A. De Luca, G. Oriolo, and B. Siciliano, “Robot redundancy resolution at the acceleration level,” *Laboratory Robotics and Automation*, vol. 4, no. 2, pp. 97–106, January 1992.

[16] M. J. Zeestraten, A. Pereira, M. Althoff, and S. Calinon, “Online motion synthesis with minimal intervention control and formal safety guarantees,” in *Proc. IEEE Sys., Man. Cybern.*, 2016, pp. 2116–2121.

[17] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, “Learning physical collaborative robot behaviors from human demonstrations,” *IEEE Trans. on Robotics*, vol. 32, no. 3, pp. 513–527, June 2016.

[18] M. J. A. Zeestraten, I. Havoutis, S. Calinon, and D. G. Caldwell, “Learning task-space synergy controllers from demonstration,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, September 2017, pp. 73–78.

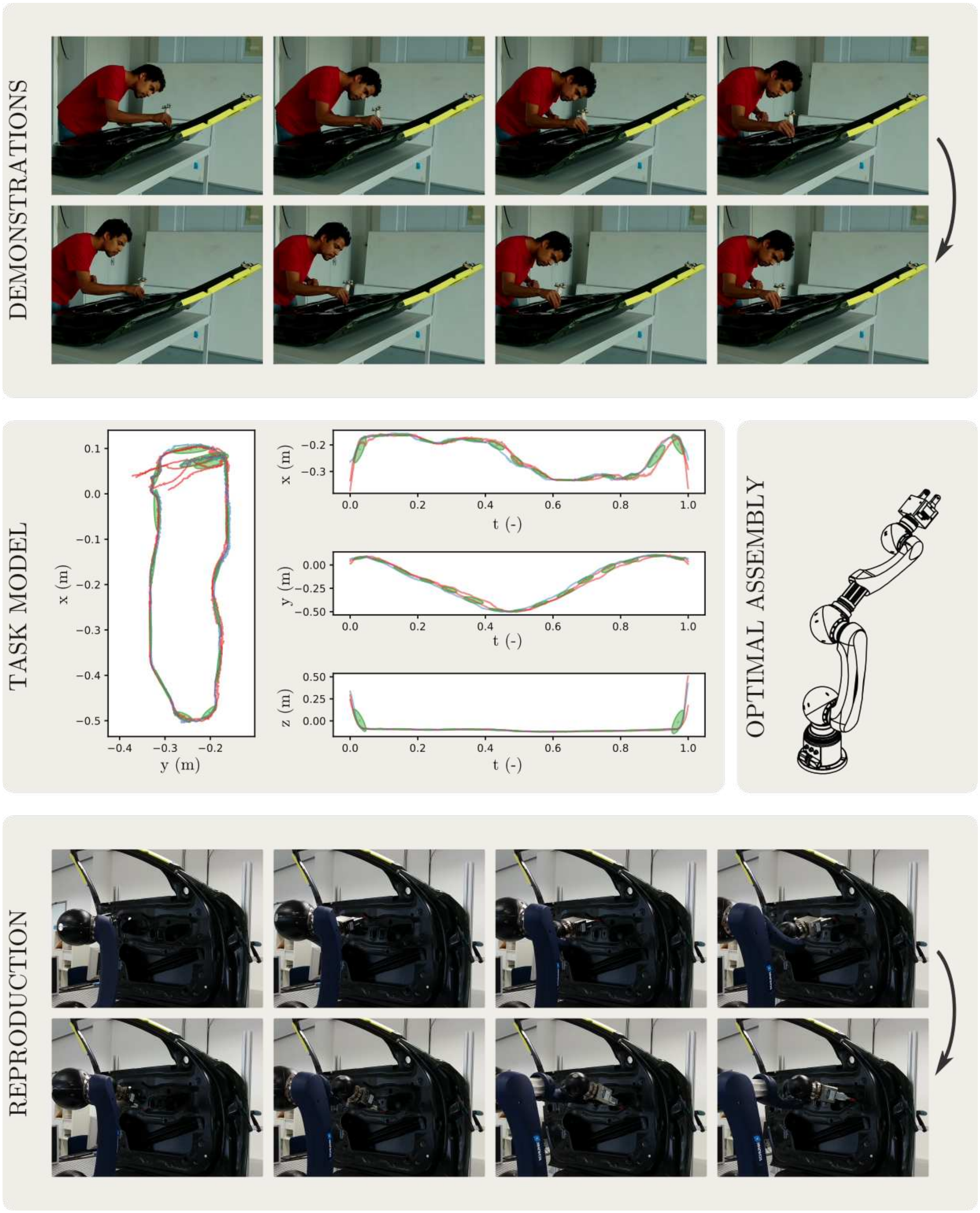


Fig. 6: Overview of experimental results for trajectory tracking task.

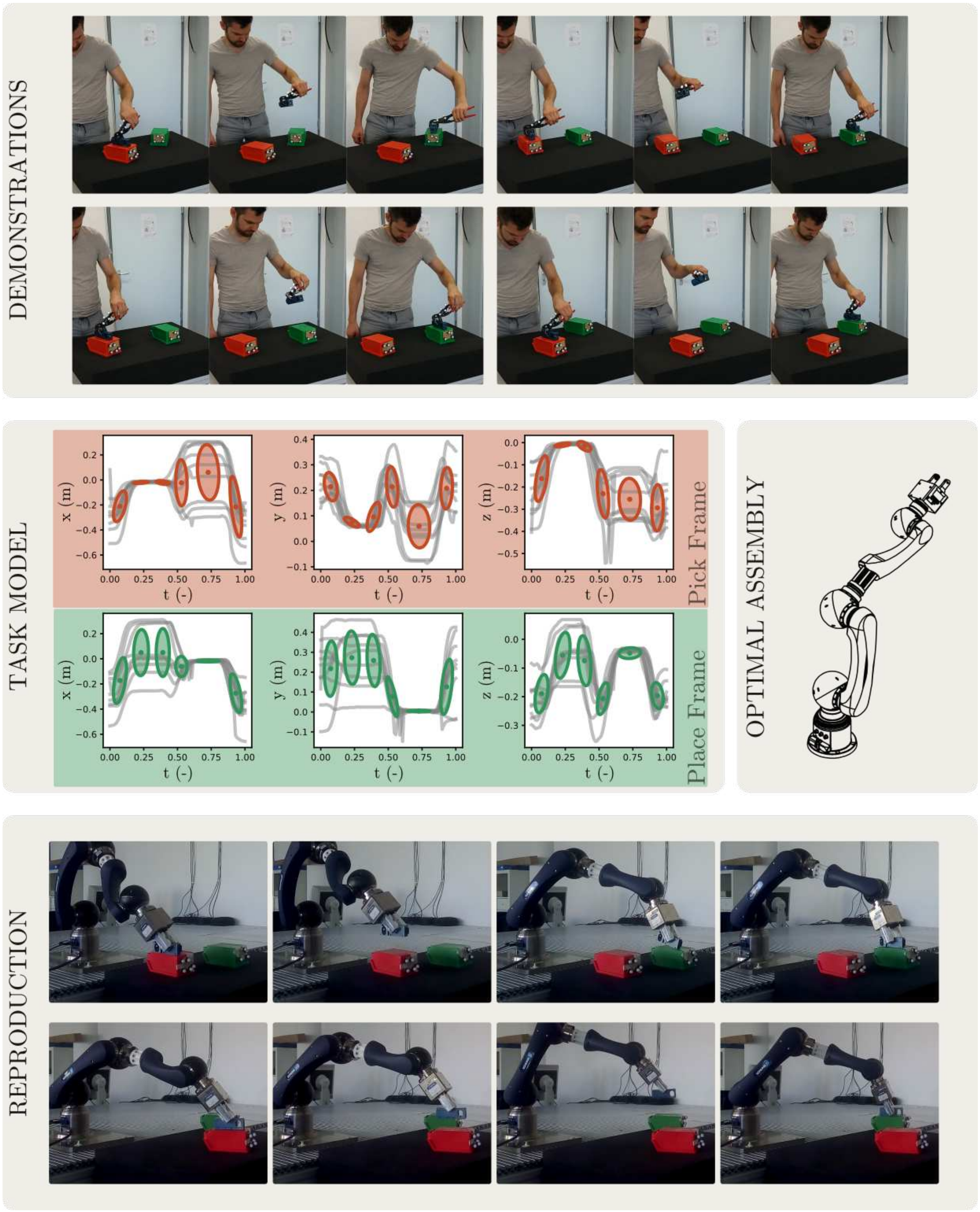


Fig. 7: Overview of experimental results for the P&P task.