2022 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
Santa Maria da Feira, Portugal – April 29-30, 2022

2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) | 978-1-6654-8217-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICARSC55462.2022.9784775

Robotic Framework for Autonomous Assembly: a Report from the Robothon 2021 Grand Challenge

Jonas Wittmann
Chair of Applied Mechanics
TUM School of Engineering and Design
Munich Institute of Robotics and Machine Intelligence
Technical University of Munich
jonas.wittmann@tum.de

Florian Pachler
Chair of Applied Mechanics
TUM School of Engineering and Design
Munich Institute of Robotics and Machine Intelligence
Technical University of Munich
florian.pachler@tum.de

Patrick Ruhkamp
Computer Aided Medical Procedures & AR
Department of Informatics
Technical University of Munich
p.ruhkamp@tum.de

Hyunjun Jung
Computer Aided Medical Procedures & AR
Department of Informatics
Technical University of Munich
hyunjun.jung@tum.de

Felix Sygulla
Chair of Applied Mechanics
TUM School of Engineering and Design
Munich Institute of Robotics and Machine Intelligence
Technical University of Munich
felix.sygulla@tum.de

Daniel Rixen
Chair of Applied Mechanics
TUM School of Engineering and Design
Munich Institute of Robotics and Machine Intelligence
Technical University of Munich
rixen@tum.de

***Abstract*—Up to now, many repetitive industrial processes, like welding in automotive, have been automated. However, robust solutions for assembly operations that, besides precise robot positioning, also require precise tactile interaction are not available yet. The current research efforts on sensitive collaborative robots and their control are a promising step towards assembly automation. In our contribution we present an autonomous robotic framework to automate assembly tasks. It implements state-of-the-art algorithms for object localization, motion planning, force control, error handling and task scheduling. We present the implementation details of our localization and insertion skills on a *FRANKA EMIKA* robot and an *Intel RealSense*, the two main components of the framework. Further, we introduce the *Robothon Grand Challenge*, a new biennial competition series that deals with automated electronic waste decomposition. Within that competition, our framework made the fifth place out of nine teams.**

***Index Terms*—Compliant Assembly, Intelligent and Flexible Manufacturing, Software Architecture for Robotic and Automation**

I. Introduction

The current consumer demand for individualized products drives factories into low-batch production. Further, due to cost pressure, more and more industrial assembly processes have to be automated. The individualized production and the need for sensitivity requires advanced technologies, such as tactile robots and robust object detection. At the same time, automation solutions need to be easy-to-use and easy-to-program for shorter system down times and changeover times, and to also enable application of such technologies in small and medium-sized enterprises without expertise in robotics.

Fig. 1. 3T robot control architecture: High-level action planning is part of the *Deliberator* that triggers a task, e.g. task board disassembly. The *Sequencer*, e.g. implemented as behavior tree, ensures the proper task execution by selecting appropriate skills from the low-level *Behavioral Control*.

The three-tiered (3T) architecture is the most dominant for the implementation of the underlying control architecture [1]. It ensures that the main system components, e.g. vision system, high-level task planner and low-level control, can interact dynamically for a robust process automation in dynamic environments. To increase the flexibility of the architecture, the current trend of robot software design goes towards robotic *skills*. These are implemented in the lower *Behavioral Control* layer of the 3T architecture which in this context is referred to as *skill-based architecture* [2]–[7]. Fig. 1 shows a schematic overview. Within our proposed robotic assembly framework, we deal with the second and third layer, i.e. the *Sequencer*

978-1-6654-8217-2/22/$31.00 ©2022 IEEE

and the *Behavioral Control*. Therefor, we implement the *Task Scheduler* as the *Sequencer* and the *Parameterizable Compliant Assembly Controller (PCAC)* and the *Board Detector (BD)* as the low-level skills. Implementation details of the three components are given in Section III.

Skills are generic preprogrammed software modules that encapsulate the expert knowledge of the robot developer and allow autonomous task planning within the 3T architecture [4]. Their parameterization must be straightforward, e.g. for the shop floor user. Robotic skills are *situated* meaning their execution is triggered in a specific system state. Further, skills can monitor their execution success using pre- and postcondition checks which is referred to as *cognizant failure* [1]. Several subfeatures, i.e. *primitives*, compose their execution process [3]. Depending on the task domain, i.e. whether the robot is applied for logistics, assembly, inspection etc. a limited set of generic robotic skills is sufficient for achieving the tasks within that domain. E.g. [7] derives six robotic skills like *pickup* or *locate* that are sufficient for logistic tasks and ten generic skills required for assistive tasks. [8] states that for assembly applications 40% of all tasks are insertion, i.e. peg-in-hole tasks.

We present the hardware and software architecture of our robot framework that automates assembly tasks. Besides motion planning, task scheduling and error handling, the localization and insertion skills are the two main components. The localization task involves the computation of the 6D pose of an object of interest in a defined coordinate frame. The insertion task is a typical peg-in-hole task in automated assembly, like key insertion or plugging a receptacle. We give the implementation details of the framework, and we demonstrate the skills' and the framework's robustness in experiments. Further, we benchmarked our framework within the *Robothon 2021 Grand Challenge* [9], a new biennial competition series that deals with automated electronic waste decomposition. We provide our results and we make our code open-source to reduce the entry barrier of the competition for future teams.

The remainder of this paper is organized as follows: In Section II, we present related work. Details on the hardware and software architecture of our framework and the localization and insertion skills are given in Section III. Section IV evaluates the skills and framework in experiments and presents the competition result. Conclusions follow in Section V.

II. Related Work

There is a big variety of solution approaches for solving the individual tasks that compose an automated assembly application. Apart from specialized solutions, the majority of the approaches are derivatives of a generic set of localization and insertion skills. [10], [11] propose frameworks that combine the localization and insertion skills for autonomous robot manipulation.

For planar objects, [12] proposes a classical localization approach by combining RGB and geometric information from an RGB-D sensor. [13] uses the fully supervised learning-based approach, and photo-realistic renderings are used to reduce the domain gap between the training and inference. [14] introduces a differentiable Perspective-n-Point (PnP) algorithm to facilitate fully end-to-end learning [15], and focus on photometrically challenging objects [16] with polarimetric information and physical cues.

[17] proposes a compliant motion control policy as the foundation for robot insertion skills. The compliant motion is implemented based on a hybrid position/force control that tries to follow a computed motion in the end effector's direction while minimizing resulting forces perpendicular to the motion. In contrast to our approach, also the target location during an insertion task is one of the inputs to the control scheme. However, this location may not always be known accurately enough or the geometric dimensions might be too small to allow position-based approaches. [18] proposes to utilize compliant control strategies that are parameterized by high-level task planners for wiping tasks. Thereby, the Cartesian motion, stiffness and force are derived as the main control parameters. Similar to our approach, [19] uses visual and tactile sensing to solve a peg-in-hole task. However, the focus lies on adaptive assembly skills, i.e. the incorporation of uncertainties due to e.g. unknown part motions using a Bayesian state estimator and an adaptive robot motion generator. [2] implements a screwing skill and a pick and place skill. However, a vision system is not incorporated. Therefore, the parameterization process of the pick and place skill, i.e. the required poses, have to be recorded from user demonstration. [6] proposes an assembly skill for mounting an antenna on an electronic board. However, only feature detection is used to compensate for position inaccuracies, and corresponding evaluations are not provided. We instead rely on Cartesian compliance and a wiggle motion as a search approach.

There are commercial app stores that provide preprogrammed robotic skills. For example in the *FRANKA WORLD* app store [20] the application *Press Button* is offered that can be parameterized with a push duration and a desired force. However, these app-based programs are less flexible as we will show in Section IV.

III. Autonomous Robotic Assembly Framework

This section gives an overview of our proposed assembly framework. We present the hardware and software architecture and the implementation details of the localization and insertion skills. Further, we introduce a new biennial robotic competition series in which we benchmarked our framework against solutions of other teams.

The design objectives of our framework are the following:

- Flexibility: Implementation of generic robotic skills that are easy to adapt to different tasks.
- Autonomy: Autonomous task scheduling that allows random object poses and includes error detection and recovery without user interaction.
- Low-cost: Exploitation of intrinsic robot sensor information and low-cost vision hardware. Total system cost: ~23.000 EUR.

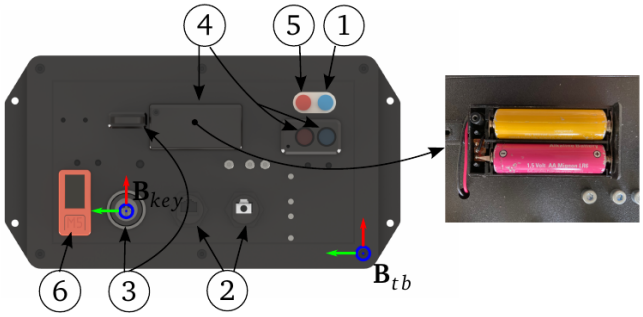

Fig. 2. Rendered view of the competition board with five assembly tasks: push two buttons (1,5); remove an Ethernet cable from a socket and plug it into another (2); grasp a key, insert it into a hole and turn it (3); remove two batteries from a case and put them in a storage (4). A microcontroller (6) was used to monitor the performance.

- Accessibility: Usage of state-of-the-art tools and open-source third-party libraries.

We developed the framework for the *Robothon 2021 Grand Challenge* competition [9], a biennial competition coinciding with the *automatica* trade fair to regularly benchmark performance in robotic manipulation. The organizing committee defines relevant tasks representing pressing and unsolved challenges in industry. In 2021, the competition took place for the first time with the focus identifying manipulation skills for electronic waste disassembly. Nine teams were selected to develop a fully automated solution using a single arm robotic manipulator. Thereby, each team was provided with the task board shown in Fig. 2 that includes five assembly tasks and that is randomly positioned in front of the robot. Detailed information on the competition and the task board are given in [9].

A. Hardware Architecture

Fig. 3 shows the hardware architecture of our approach. With its integrated torque sensors, the *FRANKA EMIKA* robot allows to implement tactile algorithms for insertion tasks with position inaccuracies. We mounted an *Intel RealSense D435i* camera on the end effector for the task board detection. The end effector is equipped with 3D-printed fingers whose design was iteratively optimized for the competition. PETG was used due to its flexibility, and the fingers are covered by a rubber coating to achieve sufficient grip for the interaction tasks. A recess is added to the finger tips for a robust grasp of the batteries and to passively ensure the correct alignment between fingers and batteries. A small extruded cylinder at the finger tip levers the batteries during removal.

B. Software Architecture

Fig. 4 shows the software architecture of the proposed robotic assembly framework that is implemented in C++. The *Robot Operating System* (ROS) is used as communication platform between the three main modules that are implemented as independent ROS nodes. For a proper interface design, ROS-provided services are used for the communication. The *Task*

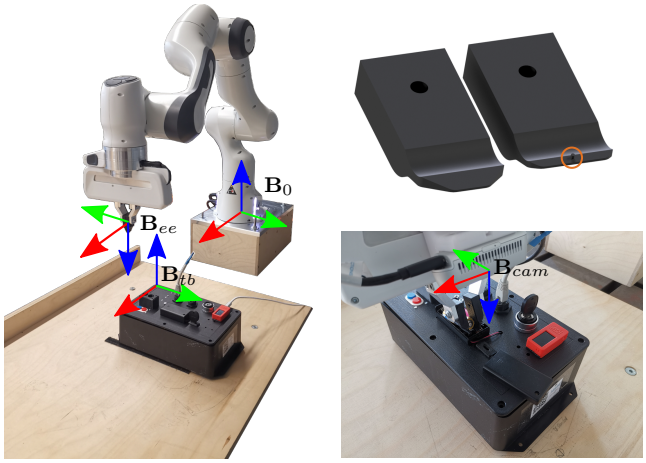

Fig. 3. Hardware architecture of our solution approach. The task board is randomly located on the surface and fixed with Velcro strips. We plan the motions of the end effector frame B_{ee} in the robot's base frame B_0. The task board pose is determined in the camera's frame B_{cam}.

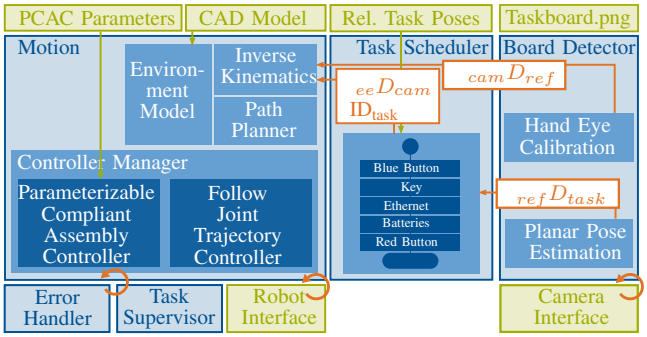

Fig. 4. The framework is composed of three main modules: *Motion*; *Task Scheduler* and *Board Detector*. The upper green boxes define the user interfaces. The lower green boxes define the hardware interfaces.

Scheduler triggers the execution of the five assembly tasks given in Fig. 2. Currently, the *Task Scheduler* is implemented as a script based on if-else-statements. The *Board Detector* implements the localization skill that detects the 6D pose of the task board (see Section III-C). The *Motion* module implements motion planning and execution, the insertion skill (see Section III-D), the *Error Handler* and the *Task Supervisor* for the assembly operation.

The upper green boxes in Fig. 4 show the four user interfaces of the framework: 1) the *PCAC Parameters* defines the task-specific parameterization of the *PCAC* (see Section III-D); 2) the *CAD Model* defines the geometric information of the environment model that is composed of simplified geometries, e.g. boxes and spheres; 3) the *Rel. Task Poses* define the relative poses of the tasks w.r.t. to the task board reference frame B_{tb} (see Fig. 2); 4) the *TaskBoard.png* image of the task board is the reference for the localization. 1)-3) are implemented as YAML files, and the values are loaded on the ROS parameter server during program start-up. The two controllers in the *Motion* module are implemented as *ROS Controllers* [21] and command the reference joint

values to the *FRANKA EMIKA* robot via the robot's hardware interface, i.e. the *franka_hw* [22]. We use [23] as the camera interface.

The *Task Scheduler* calls the services provided by *Motion* and *Board Detector* to implement the following high-level process in which *ID* denotes a specific task, e.g. *key*:

1) Generate environment model
2) Detect task board location
3) For each task:
   a) Move the end effector to task pose ${}_0\mathbf{D}_{ID}$
   b) Parameterize and activate PCAC

While the task board's geometry is predefined, its location is not. The required end effector pose to solve a task *ID* w.r.t. the task board's reference frame B_{tb} is defined by the user within the *Rel. Task Poses* file and given by ${}_{tb}\mathbf{D}_{ID}$. The transformation between the camera frame and the end effector frame ${}_{ee}\mathbf{D}_{cam}$ is determined by a hand-eye calibration (see Section III-C). With the online localization of the task board frame w.r.t. the camera frame ${}_{cam}\mathbf{D}_{tb}$ the following transformation determines the required end effector pose w.r.t. robot's base frame B_0 for each task:

$$ {}_0\mathbf{D}_{ID} = {}_0\mathbf{D}_{ee}\ {}_{ee}\mathbf{D}_{cam}\ {}_{cam}\mathbf{D}_{tb}\ {}_{tb}\mathbf{D}_{ID} \tag{1} $$

To plan and execute the motion of the end effector, *Inverse Kinematics* on position-level are computed for the end effector's goal ${}_0\mathbf{D}_{ID}$ within the *Path Planner*. The implementation of the *RRT-connect* planner [24] in the open-source framework *MoveIt!* [25] plans collision-free motions in the robot's configuration space based on the *Environment Model*, and the computed path is tracked by the *Follow Joint Trajectory Controller*.

The *Task Supervisor* monitors the task success, e.g. based on the achieved push force when pressing the button. Due to inaccuracies during object detection and positioning of the end effector the success rate of the insertion tasks, esp. the key insertion, limits the system's robustness. Therefore, we implemented the *Error Handler* that monitors the execution of insertion tasks and triggers retries in case of failure. Algorithm 1 shows the implemented logic. To avoid deadlocks during the insertion, we define the timeout t_{lim}. The insertion process is defined successful when a position threshold z_{lim} is reached. After each failed insertion attempt, the pre-insertion pose of the end effector w.r.t. B_{ID} is modified: a task-independent delta, whose magnitude depends on the current number of attempts n, is added to the x-coordinate. We only modify the x-component as pose estimation inaccuracies in this direction were highest in tests and had great impact on the robustness, especially for the key insertion. This is due to inaccuracies when estimating the orientation of B_{tb} in Fig. 2 that, due to the considerable y-distance of B_{tb} and B_{key}, lead to a reasonable inaccuracy in the x-component of B_{key}. We define a maximum number of attempts n_{max}.

Algorithm 1 Error Handling for Insertion Skills

```
n ← 0
while (n < n_max) do
    moveEndEffector(0D_ID)
    while t < t_lim do
        if (z_ee < z_lim) then
            return true
        end if
    end while
    n ← n + 1
    adaptPreInsertionPose(n)
end while
return false
```

C. Localization: Board Detector

Here, we implement two different approaches, a learning-based and a classical approach, to solve the task of 6D object pose estimation of the task board in the camera coordinate system B_{cam}. To transform the pose to the coordinate system of the end effector B_{ee}, a hand-eye calibration [26] is performed.

For the learning-based pipeline, we leverage [13], a two step approach composed of heat map prediction and pose refinement, which is trained in a supervised fashion. A CNN first predicts eight corner points of the task board in the given image, then the final pose is computed through EPnP [27] with predefined corner points on the 3D CAD model of the object. To provide a realistic and robust data set for training, we rendered the CAD model of the task board in *Unity* with various augmentations such as material, lighting, camera pose and backgrounds. The training is done with a total of 7500 images and took 14 epoch to converge.

For the classical approach, we implement the pose estimation approach in [12] as the task board has a planar surface area and leverage the RGB-D information of our setup. To detect the pose ${}_{cam}\mathbf{D}_{tb}$ of the planar task board, we need to detect three orthonormal vectors on the surface that describe the coordinate frame of the object. Therefore, the robot first moves to a predefined pose above the expected task board area and acquires an overhead image. Robust features [28] on the task board of the live camera view are matched with a pre-acquired user-provided target image (*Taskboard.png* in Fig. 4). A homographic transformation between the current view and the target is computed from the feature matches with a robust *RANSAC* scheme to filter outliers. The target image defines three points on the image plane, with a center point and two points with orthogonal vectors (e.g. the image center and two points in x- and y-axis of the image, respectively). These points are transformed into the current view via the estimated homographic transformation. Now, the depth values from the active depth sensor are queried from the point cloud and the 3D center point yields the translational part of the task board pose. The z-axis (towards the camera) is computed as cross product of the x- and y-axis, where the axes are defined by the three coplanar points defined before. The rotation is then computed from the unit vectors along the three axes.

D. Insertion: Parameterizable Compliant Assembly Controller

We rely on an impedance-based insertion strategy similar to [29]. However, instead of adding an additional wrist force sensor, we use the integrated torque sensors. We implemented the *PCAC* as our insertion skill that computes the reference joint torques $\boldsymbol{\tau}_{ref}$ in each 1 kHz control cycle. Our implementation builds upon the algorithm in [30] that implements a Cartesian force controller and a Cartesian impedance controller. Compared to [30], we add a D-gain and I-gain for the force controller and the impedance controller respectively. Further, our implementation allows force control in arbitrary directions which is required e.g. for turning the key or removing the batteries. Algorithm 2 shows our implementation of the *PCAC*. Therein, $\mathbf{x}_{EE}$ and $\mathbf{p}_{EE}$ define the end effector's position and orientation respectively. Note that orientation computations are implemented using unit quaternions and $\otimes$ defines the quaternion multiplication. $\mathbf{f}_{EE}$ defines the external forces and torques acting on the end effector. The *PCAC* composes three primitives: a PID Cartesian force controller (line 4); a Cartesian impedance controller (line 12) and a wiggle motion feature (line 8). The former applies the parameterized forces, e.g. for pressing the button, in the defined coordinate direction. As no force trajectories are computed and as the force controller has to be activated when there is no contact with the environment yet, the PID control scheme ensures a smooth application of the desired force value (instead of e.g. using only a P gain). The Cartesian impedance controller keeps the end effector at the current pose, e.g. during turning the key or when removing the Ethernet plug. The wiggle motion compensates position errors during insertion tasks which set the *insertionTask* flag to activate the wiggle motion feature within the *PCAC*. Note that in the sense of a hybrid position/force control, one coordinate direction is either controlled by the Cartesian impedance control or by the PID force control. This is enabled within the *PCAC Parameters* file (see Fig. 4) by setting either the desired force or the desired Cartesian stiffness to zero for the respective coordinate direction.

Within the *PCAC Parameters* file in Fig. 4, the user defines the specific parameters for each task: desired end effector force $\mathbf{f}_{EE,des}$; P-, I- and D-gains for the force controller and impedance controller; flag *insertionTask* to activate the wiggle motion; frequency f and amplitude A of the wiggle motion. The initial end effector position $\mathbf{x}_{EE,init}$ and the initial end effector orientation $\mathbf{p}_{EE,init}$ are stored at controller start-up and further serve as input. To ensure critical damping, we set $\mathbf{K}_{D,Imp} = \sqrt{2}\mathbf{K}_{P,Imp}$. A second-order low pass filter processes the force error and computes the corresponding D-component of the PID force controller. The wiggle motion applies a sinusoidal oscillation of the end effector's orientation in its x- and y-direction based on f and A. The dynamic robot model computes the compensation of the gravity, centrifugal and Coriolis forces $\mathbf{g}(\mathbf{q})$ and $\boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}})$.

Algorithm 2 Insertion Skill

Input: $\mathbf{x}_{EE,init}, \mathbf{p}_{EE,init}, \mathbf{f}_{EE,des}, \mathbf{K}_{P,PID}, \mathbf{K}_{I,PID}, \mathbf{K}_{D,PID}, \mathbf{K}_{P,Imp}, \mathbf{K}_{I,Imp}, f, A$, insertionTask
Output: $\boldsymbol{\tau}_{ref}$

1: $\mathbf{f}_{err} \leftarrow \mathbf{f}_{des} -$ estimateExternalForce()
2: $\mathbf{f}_{err,I} \leftarrow \int \mathbf{f}_{err}\, dt$
3: $\mathbf{f}_{err,D} \leftarrow$ secondOrderLowPassDerivative($\mathbf{f}_{err}$)
4: $\mathbf{f}_{PID} \leftarrow \mathbf{K}_{P,PID}\mathbf{f}_{err} + \mathbf{K}_{I,PID}\mathbf{f}_{err,I} + \mathbf{K}_{D,PID}\mathbf{f}_{err,D}$
5: $\mathbf{x}_{EE,err} \leftarrow \mathbf{x}_{EE,init} - \mathbf{x}_{EE}$
6: $\mathbf{p}_{EE,des} \leftarrow \mathbf{p}_{EE,init}$
7: **if** insertionTask **then**
8: $\quad \mathbf{p}_{EE,des} \leftarrow$changeOrientation($f, A, \mathbf{p}_{EE,init}$)
9: **end if**
10: $\mathbf{p}_{EE,err} \leftarrow \mathbf{p}_{EE}^{-1} \otimes \mathbf{p}_{EE,des}$
11: $\boldsymbol{\omega}_{err} \leftarrow [\mathbf{x}_{EE,err}; \mathbf{p}_{EE}.\text{toRotMatrix}()\log(\mathbf{p}_{EE,err})]$
12: $\mathbf{f}_{Imp} \leftarrow \mathbf{K}_{P,Imp}\boldsymbol{\omega}_{err} + \mathbf{K}_{I,Imp}\int \boldsymbol{\omega}_{err}\, dt - \mathbf{K}_{D,Imp}\mathbf{J}\dot{\mathbf{q}}$
13: $\boldsymbol{\tau}_{ref} \leftarrow \mathbf{g}(\mathbf{q}) + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^T(\mathbf{f}_{PID} + \mathbf{f}_{Imp})$

TABLE I
TASK BOARD POSE ESTIMATION ERROR

	Translational Error	Rotational Error
RGB-D	9.8 ± 3.5 mm	**0.72**$\pm 0.32°$
Learning-Based	**9.2**± 3.8 mm	$7.32 \pm 4.46°$

IV. RESULTS

A. Board Detector

We benchmark the accuracy of our localization and compare the learning-based approach [13] against the classical method [12]. Table I shows the average pose error and the standard deviation for both approaches over eight trials. The pose error is computed against the ground-truth pose of the task board, which is determined by measuring the 3D coordinates of four corner points of the task board through the forward kinematics of the robot's end effector. The rotational component of the estimated 6D pose for the learning-based approach is less accurate with large deviations. Hence, we opted in our final system to use the classical approach due to its better performance, generalization and no need for training. The relative large translational error is due to inaccuracies of the depth sensor which lead to errors in the z-direction of the estimated position. For this reason, the height of the task board is hard coded, as the board usually lies on a flat surface with constant height w.r.t. the robot base. The proposed insertion skill in Section III-D was able to compensate the translational error and successfully achieved the insertion tasks.

B. Robotic Assembly Framework

To evaluate the robustness of our framework, we did ten task board runs. Fig. 5 shows a complete task board run. We benchmark the *PCAC* implementation with a human and an easy-to-program app-based solution based using the *FRANKA WORLD* app store [20] (see Section I). Table II compares the task-specific average success rate, the average solution time

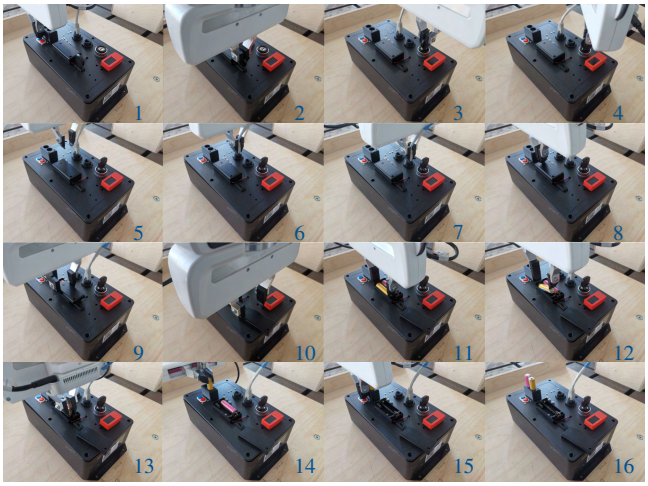

Fig. 5. Complete task board run with the five assembly tasks in Fig. 2.

TABLE II
BENCHMARK OF TASK BOARD SOLUTION APPROACHES

	Success Rate [%]				Time [s]	Effort	Flex.
	Button	Key	LAN	Battery			
Human	100	100	100	100	16	•	•••
Framework	100	90	100	85	153	•••	••
App	100	100	100	75	193	••	•

for the task board and the qualitative implementation effort and flexibility of the three approaches.

The ten task board runs include 20 button presses (red and blue), ten key insertions, ten Ethernet insertions and 20 battery removal and insertions operations (two batteries). The success rate is determined as the number of successful task executions divided by the number of total task attempts, e.g. 20 for the button task. Both automation solutions can not compete with a human worker. The app-based approach and our framework showed similar robustness for the individual tasks. Both can solve tactile interaction tasks like pressing a button robustly. Also insertion tasks with a lower demand on position accuracy, e.g. the LAN task, are solved robustly. Insertion tasks with a higher demand on position accuracy, e.g. the key insertion, are more challenging, and our framework failed once in the evaluation. Thereby, our error handler adapted the key insertion pose in seven runs. The battery recycling is composed of several subtasks and therefore showed the worst performance for both automation solutions. W.r.t. time our framework can compete with the app-based program and even shows better performance. However, note that this is a qualitative statement as we did not exploit the actuators' limits. When it comes to implementation effort, our framework is worse than the app-based approach. The main advantage of our framework compared to the app-based approach is its flexibility: the app-based approach restricts the integration of further modules and therefore can not be used for applications in which the task board pose is not known. In contrast to that, our framework allows a localization of the task board and is thus able to solve it in various pose ranges. The corresponding evaluation video shows all three approaches and the ten runs of our framework with random repositioning of the task board: https://youtu.be/Yg3tPoy7zxw. Note that we did the evaluation with the current implementation that we have worked on since the *Robothon 2021 Grand Challenge*.

With a previous version of our proposed framework, we made the fifth place in the *Robothon 2021 Grand Challenge* competition and the results are available at [9]. During the competition we could resolve all tasks except turning the key and took 200 seconds.

V. CONCLUSIONS

In contrast to welding and drilling, robust automation solutions for industrial assembly operations are not mature yet. Within these applications, there is still a lack of technology transfer from state-of-the-art research to industry. The proposed framework demonstrates the potential of low-cost automation solutions for assembly tasks. The main features of the individual software modules are based on open source third-party libraries. The proposed localization is accurate enough for a robust execution of the other robotic tasks, but is limited to piece-wise planar objects like the task board we used. Compared to that, a learning-based approach suffers from large rotational errors, which may be accounted for in the future with a self-supervised 6D pose refinement stage. Such approaches are an active and challenging field of research in the vision community. The proposed insertion approach, combined with the proposed error handling, shows a robust operation. Its straightforward parameterization allows its application in a wide range of assembly and tactile interaction operations. To lower the entry barrier of the biennial *Robothon Grand Challenge* competition series for other teams, we make our framework open source: https://gitlab.lrz.de/AM/rgc21. We also provide the files for our 3D-printed fingers and the file for running the app-based program using *FRANKA Desk*.

REFERENCES

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[2] F. Steinmetz and R. Weitschat, "Skill parametrization approaches and skill architecture for human-robot interaction," *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 280–285, 2016.

[3] M. Pedersen, L. Nalpantidis, R. Andersen, C. Schou, S. Bgh, V. Krger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, 02 2016.

[4] M. Crosby, F. Rovida, M. Pedersen, R. Petrick, and V. Krüger, "Planning for robots with skills," in *Proceedings of the 4th Workshop on Planning and Robotics (PlanRob)*. ICAPS, Jun. 2016, pp. 49–57, 4th ICAPS Workshop on Planning and Robotics 2016, PlanRob 2016.

[5] J. Saukkoriipi, T. Heikkilä, J. Ahola, T. Seppälä, and P. Isto, "Programming and control for skill-based robots," *Open Engineering*, vol. 10, no. 1, pp. 368–376, Jan. 2020.

[6] H. Herrero, A. Abou Moughlbay, J. Outn, D. Salle, and K. Lopez-de Ipia, "Skill based robot programming: Assembly, vision and workspace monitoring skill interaction," *Neurocomputing*, vol. 255, 03 2017.

[7] S. Bøgh, O. Nielsen, M. Pedersen, V. Krüger, and O. Madsen, "Does your robot have skills?" in *Proceedings of the 43rd International Symposium on Robotics*. VDE Verlag GMBH, Aug. 2012.

[8] J. Jiang, Z. Huang, Z. Bi, X. Ma, and G. Yu, "State-of-the-art control strategies for robotic pih assembly," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101894, 2020.

[9] Munich Institute of Robotics and Machine Intelligence. (2021) Robothon- the grand challenge series of munich_i an international competition in robot manipulation. [Online]. Available: https://www.robothon-grand-challenge.com/

[10] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A framework for robot manipulation: Skill formalism, meta learning and adaptive control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[11] D. Wahrmann, A.-C. Hildebrandt, C. Schuetz, R. Wittmann, and D. Rixen, "An autonomous and flexible robotic framework for logistics applications," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 3, pp. 419–431, Mar 2019.

[12] S. K. Paul, M. T. Chowdhury, M. Nicolescu, and M. Nicolescu, "Object detection and pose estimation from rgb and depth data for real-time, adaptive robotic grasping," in *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, 2020.

[13] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Conference on Robot Learning (CoRL)*, 2018.

[14] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, "Ffb6d: A full flow bidirectional fusion network for 6d pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[15] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 16 611–16 621.

[16] D. Gao, Y. Li, P. Ruhkamp, I. Skobleva, M. Wysock, H. Jung, P. Wang, A. Guridi, N. Navab, and B. Busam, "Polarimetric pose prediction," 2021.

[17] F. Dai, A. Wahrburg, B. Matthias, and H. Ding, "Robot assembly skills based on compliant motion," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, 2016, pp. 1–6.

[18] D. Leidner, A. Dietrich, M. Beetz, and A. Albu-Scheffer, "Knowledge-enabled parameterization of whole-body control strategies for compliant service robots," *Autonomous Robots*, 03 2016.

[19] K. Nottensteiner, A. Sachtler, and A. Albu-Scheffer, "Towards autonomous robotic assembly: Using combined visual and tactile sensing for adaptive task execution," *Journal of Intelligent & Robotic Systems*, vol. 101, 03 2021.

[20] FRANKA EMIKA GmbH. (2021) Franka world. [Online]. Available: https://world.franka.de/products

[21] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017.

[22] FRANKA EMIKA. (2022) franka_hw. [Online]. Available: https://github.com/frankaemika/franka_ros/tree/develop/franka_hw

[23] Intel RealSense team. (2022) Ros wrapper for intel realsense devices. [Online]. Available: https://github.com/IntelRealSense/realsense-ros

[24] J. Kuffner and S. 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 vol.2.

[25] "MoveIt!" https://moveit.ros.org/, accessed: 2021-06-12.

[26] R. Tsai and R. Lenz, "Real time versatile robotics hand/eye calibration using 3d machine vision," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 554–561 vol.1.

[27] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.

[28] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, 11 2004.

[29] P. Zou, Q. Zhu, J. Wu, and J. Jin, "An approach for peg-in-hole assembling based on force feedback control," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 3269–3273.

[30] "Force control tutorial by FRANKA EMIKA," https://github.com/frankaemika/icra18-fci-tutorial, accessed: 2021-06-12.