IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

RAMPA: Robotic Augmented Reality for Machine Programming by DemonstrAtion

Fatih Dogangun[1], Serdar Bahar[1], Yigit Yildirim[1], Bora Toprak Temir[1], Emre Ugur[1], and Mustafa Doga Dogan[2,1]

***Abstract*—This paper introduces Robotic Augmented Reality for Machine Programming by Demonstration (RAMPA), the first ML-integrated, XR-driven end-to-end robotic system, allowing training and deployment of ML models such as ProMPs on the fly, and utilizing the capabilities of state-of-the-art and commercially available AR headsets, e.g., *Meta Quest 3*, to facilitate the application of Programming by Demonstration (PbD) approaches on industrial robotic arms, e.g., *Universal Robots UR10*. Our approach enables *in-situ* data recording, visualization, and fine-tuning of skill demonstrations directly within the user's physical environment. RAMPA addresses critical challenges of PbD, such as safety concerns, programming barriers, and the inefficiency of collecting demonstrations on the actual hardware. The performance of our system is evaluated against the traditional method of kinesthetic control in teaching three different robotic manipulation tasks and analyzed with quantitative metrics, measuring task performance and completion time, trajectory smoothness, system usability, user experience, and task load using standardized surveys. Our findings indicate a substantial advancement in how robotic tasks are taught and refined, promising improvements in operational safety, efficiency, and user engagement in robotic programming.**

***Index Terms*—Virtual Reality and Interfaces, Learning from Demonstration, Augmented Reality (AR), Extended Reality (XR)**

I. INTRODUCTION

The advent of sophisticated robotic applications necessitates advanced programming techniques that can accommodate complex, dynamic environments. The conventional approach to robot programming typically involves *segmented* processes where users must *alternate between physical and digital realms*. Traditional approaches rely heavily on offline simulation and manual coding, which can be time-consuming and often impractical for real-time applications [1]. As these methods lack the capability to adapt quickly to new environments or to intuitively incorporate human expertise, the complexity and time involved in programming increases, which also increases the risk of errors and accidents.

The procedure known as Programming by Demonstration (PbD), a.k.a. Learning from Demonstration (LfD), is a widely adopted procedure that enables robots to learn skill policies by watching an expert [2]. Recently, Machine Learning (ML)-based PbD frameworks have become instrumental in enabling robots to model and execute intricate skills autonomously. Yet, integrating such approaches into robotic programming often encounters significant challenges, including the labor-intensive and potentially hazardous processes involved in data gathering and model testing in the real world [3].

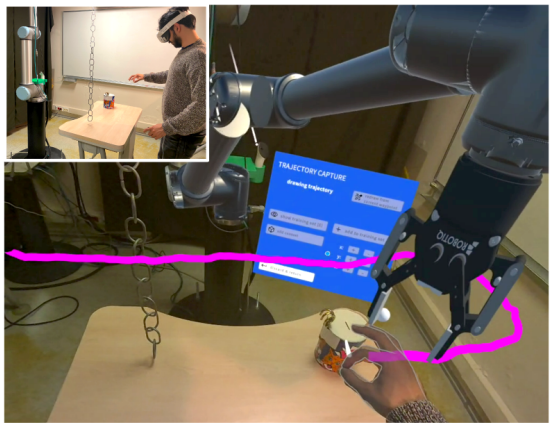

Fig. 1. With RAMPA, novice and expert users can demonstrate trajectories and simulate robot movements *in situ* using augmented reality (AR).

In response to these challenges, we introduce a sophisticated AR-based framework that transforms robotic programming through direct, interactive modeling and visualization. Our system, Robotic Augmented Reality for Machine Programming by Demonstration (RAMPA), leverages the immersive and interactive capabilities of state-of-the-art XR headsets (e.g., *Meta Quest 3*) to create a cohesive and safe user experience for robot programming and automation of tasks ***in situ***. As shown in Fig. 1, users can directly demonstrate robot trajectories and visualize robot movements while maintaining the awareness of the actual workspace. We also release RAMPA as open source for the robotics community, offering compatibility with cutting-edge mixed reality headsets and robotic arms.

By integrating in-situ simulation and direct manipulation capabilities, RAMPA offers a dual advantage: it streamlines the programming process while enhancing the physical safety, flexibility of robotic operations, and efficiency of PbD procedures and, therefore, reduces the barriers to effective human-robot collaboration. The ability to produce and adjust demonstration trajectories in real-time and in situ facilitates the task of the operator while interacting directly with the learned models establishes a deeper understanding of these models.

Our system facilitates a seamless, intuitive interface for:

a) Data Recording: Operators use XR to demonstrate skills, capturing the data directly in the user's environment. Compared to traditional methods, such as kinesthetic control, RAMPA simplifies the collection of demonstration trajectories,

This work has been supported by the *INVERSE* project (no. 101136067), funded by the EU.

[1] Department of Computer Engineering, Bogazici University, Istanbul, Türkiye. [2] Adobe Research, Basel, Switzerland. Corresponding author: fatih.dogangun@std.bogazici.edu.tr

Code is available at https://github.com/dogadogan/rampa

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

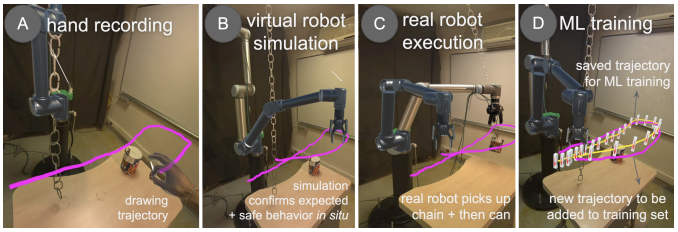

Fig. 2. RAMPA allows users to (a) draw trajectories with their bare hand, (b) simulate and observe their effect *in situ*, and (c) execute them on the real robot. (d) RAMPA facilitates the direct and streamlined collection of trajectory data to train ML models. Please refer to text for use cases of the system.

an integral component of PbD procedures, offering a more efficient experience for robotic programmers.

b) Data Playback and Adjustment: The system allows operators to visualize and modify the movements of a virtual robot before these movements are executed on the actual robotic arm. This step ensures that potential errors can be corrected in a safe, virtual space. Moreover, the ability to adjust any portion of demonstration trajectories increases the operator's overall performance.

c) Model Training and Validation: Utilizing the collected data, popular PbD models, such as Probabilistic Movement Primitives (ProMPs) [4], can be trained and validated within the AR environment. Operators can observe the generated trajectories in real-time, ensuring the robot's behavior aligns with desired outcomes.

To the best of our knowledge, we propose the first end-to-end XR-driven PbD framework to enable the demonstration of robotic trajectories, training, and testing ML model augmented by visualization of the robot motion and trajectory adjustment in situ. An overview of the proposed system can be seen in Fig. 2. Importantly, our comprehensive evaluation comprises the experiment consisting of three distinct robotic tasks, 20 participants including both experienced and novice users, and an extensive analysis using quantitative metrics, incorporating standard surveys, such as *System Usability Scale* (SUS) [5], *User Experience Questionnaire* (UEQ) [6], and *NASA Task Load Index* (NASA-TLX) [7], and qualitative user feedback.

This paper details the development and implementation of our system, evaluates its performance against the traditional method of kinesthetic control, and demonstrates its effectiveness through empirical studies. Our findings indicate a significant advancement in how robotic tasks are taught and refined, promising notable improvements in operational safety, efficiency, and user engagement in robotic programming.

In summary, we make the following contributions:

- A novel AR-based system that enables users to create and fine-tune robotic applications that mimic human behavior, allowing safe, in-situ simulation and execution.
- Comprehensive evaluation and analysis of the system in teaching different robot manipulation tasks using both qualitative feedback and quantitative metrics measuring task performance and completion time, trajectory smoothness, system usability, user experience, and task load.
- Facilitation of interactive and situated programming of PbD models suitable for commonly used movement primitives in robotics.

II. RELATED WORK

A. Movement Primitives in Programming from Demonstration

Movement primitives (MP), which provide effective abstractions that facilitate motor control [8], is a modular formalism widely employed to encode and reproduce complex movements. Schaal et al. [8] proposed the Dynamical Movement Primitives (DMPs) model to learn MPs by leveraging human expertise. The original DMP model could encode only a single trajectory, while different demonstrations may contain important variabilities that the modeling framework must capture [9], [10]. In order to capture the variety in demonstrations, Gribovskaya et al. [11] proposed to model movement trajectories using probabilistic methods. These methods allowed the use of multiple demonstrations to model the skill trajectories, enhancing the representational power and the generalization capacity of MPs [12].

Latest advances in ML have significantly influenced PbD approaches, enabling the synthesis of more flexible and generalizable MP-based controllers to perform tasks with higher efficiency and adaptability. Recent studies, such as Pérez-Dattari et al. [13], Blessing et al. [14], Yildirim et al. [15], use deep neural networks to model movement trajectories in the form of MPs. However, they are not practical for real-time applications, specifically for human-in-the-loop interactive systems, as they need a training phase to model demonstration trajectories. Due to the real-time requirement, ProMPs, which models a distribution of trajectories as a probability distribution over a set of basis functions, improving the generalization capability to novel via points [16], is used as the MP modeling framework in this study.

In RAMPA, integrating AR into the ML training and visualization process allows users to demonstrate tasks within an AR environment, eliminating manual spatial data recording. Moreover, RAMPA enhances the comprehensibility of ML models by offering a interactive platform for ML training and fine-tuning.

B. AR-Oriented Interfaces

Different types of robot programming frameworks, such as visual programming methods Paxton et al. [17] combined with speech-based programming Alexandrova et al. [18] have been proposed to allow users to communicate instructions and feedback to robots with ease and high precision, which improves the functionality and usability of robotic systems and enables efficient human-robot interaction. Recently, foundation models have been deployed in real-world robotic applications; nevertheless, they are still inadequate for tasks requiring human involvement [19].

Unlike the traditional methods, AR technologies, with their real-time and contextual feedback directly within the user's field of view, make it possible to have more natural and efficient human-robot interactions. As a subset of XR technologies, they enable the superimposition of computer graphics onto the physical world [20] and have been increasingly utilized to enhance human-robot interaction (HRI), bridging

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

the gap between virtual and real environments. Recent surveys Walker et al. [21] and Suzuki et al. [3] demonstrate the potential of AR to transform robot programming, providing a detailed taxonomy of AR applications in robotics, including beneficial aspects of AR for robot control, programming, and visualization. Specifically, Jiang et al. [22] compare and contrast multiple interaction interfaces for AR-based robot programming, highlighting how different interface designs impact data collection efficiency and overall user experience.

AR visualization of the robot's motion has been used to simplify the process of planning, observing, and modifying the robot's trajectory for users. For instance, Cao et al. [23] enables users to visualize and adjust robot trajectories in real-time, which improves the accuracy of robot actions in human-robot collaborative operations. Similarly, Quintero et al. [24] introduces a method that allows users to interact with robot trajectories with ease by using an AR interface to visualize the motion of the robot. Walker et al. [25], Rosen et al. [26] show that visualizing robot movement improves task efficiency and enhances user understanding of robot intentions, hence facilitating human-robot interactions.

AR interfaces have also been used for robot training methods such as PbD and teleoperation, as they offer interactive and intuitive ways to teach robots new skills. Arunachalam et al. [27] provides users with a broad AR-based framework for robot learning with teleoperation. Alongside, Luebbers et al. [28] offers in-situ visualizations of robot trajectories with their constraints, and Diehl et al. [29] proposes a system for visualization of learned skills for verification and error detection, therefore providing an intuitive platform for users to convey proper demonstrations to the robot. However, these studies require kinesthetically demonstrated trajectories. Lotsaris et al. [30] introduces an AR system to demonstrate robot trajectories by placing the virtual gripper in the target position. This work, however, does not allow the user to demonstrate the precise path. Recent works Chen et al. [31] and Chen et al. [32] show the potential of AR interfaces, focusing on the demonstration collection phase. However, they do not offer an end-to-end solution. By bridging the gap left by AR-based interfaces for PbD, we propose a comprehensive, *end-to-end system* that allows users to demonstrate robotic trajectory, visualize the robot motion, modify the trajectory, train, and test the ML model *in situ*, hence making robotic programming more accessible and practical.

A comprehensive comparison with existing AR-based systems is provided in Table I, evaluating several key features. The *End-to-End* feature corresponds to whether the proposed system supports demonstration collection, the training and deployment of the training result on the fly, within a single system. *ML in AR* indicates whether any ML component is integrated into the AR system. *Full Trajectory Modification* refers to whether the system has rewinding, redrawing capabilities of the collected trajectory pre- or post-demonstration by the actual robot. *Online Hand Mimicking by Robot* corresponds to whether the system enables online hand mimicking during demonstration collection by either the simulated or the actual robot. *Trajectory Visualization* represents the capability to visualize the collected trajectory within AR, where *Partial* corresponds to constraint visualization of the learned skill in Luebbers et al. [28], and end-effector visualization in Lotsaris et al. [30]. Lastly, *Trajectory Drawing* describes the method of trajectory collection within AR.

TABLE I
COMPARISON OF AR-BASED SYSTEMS

Feature / Method	[24]	[28]	[30]	[31]	[32]	RAMPA
End-to-End	No	No	No	No	No	**Yes**
ML (Training + Deployment) in AR	No	**Yes**	No	No	No	**Yes**
Full Trajectory Modification	No	No	No	No	No	**Yes**
Online Hand Mimicking by Robot	No	No	No	**Yes**	**Yes**	**Yes**
Trajectory Visualization	**Yes**	Partial	Partial	No	**Yes**	**Yes**
Trajectory Drawing Method	Point-based	None	Point-based	Hand gestures	Hand gestures	Hand gestures

III. RAMPA: AN END-TO-END SYSTEM FOR ROBOTIC AR

The RAMPA system integrates a ROS server, Unity game engine, and Quest 3 XR headset to provide an intuitive platform for interactive demonstration collection, ML training, and visualization. Its interface, implemented in Unity, include a simulated robot, a UI menu, and interactive tools such as virtual markers for model conditioning. Users control the system via hand gestures and controller inputs, leveraging features like trajectory adjustment, auto-calibration, online hand following, and ML integration.

A. System Overview

RAMPA features key functionalities to facilitate safe and cohesive trajectory capturing with hand mimicry, fine-tuning of the demonstrated trajectories, use of this recorded data to train ML models and safe execution of the recorded trajectories in the virtual environment, or the trajectories sampled from the trained model on the real robot.

1) Simulated Robot and Auto-Calibration: A simulated robot is present to model the actual robot's behavior, allowing for accurate testing and validation of trajectory plans. For deployment of demonstrations to actual robot, the simulated and the actual robot should be precisely aligned. This problem is addressed by the auto-calibration feature. The feature uses the controller from the headset and a 3D-printed tool that can hold the controller for the virtual robot to be aligned with the actual robot, as shown in Fig. 2-A. The simulation environment ensures that any trajectory can be tested thoroughly for safety before being deployed on the real robot, also making sure the actual robot will replicate both the location and the motion of the simulated one.

2) Trajectory Playback and Adjustment: Trajectory playback and adjustment are implemented to allow the user to fine-tune the recorded trajectory. When a user decides to inspect the current trajectory with the simulated robot, before or after real robot demonstration, they can go backward or forward in the trajectory, play the rest of the re-winded trajectory, and pause when desired (Fig. 3), using the UI provided in the system. If the user decides to discard a portion of the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

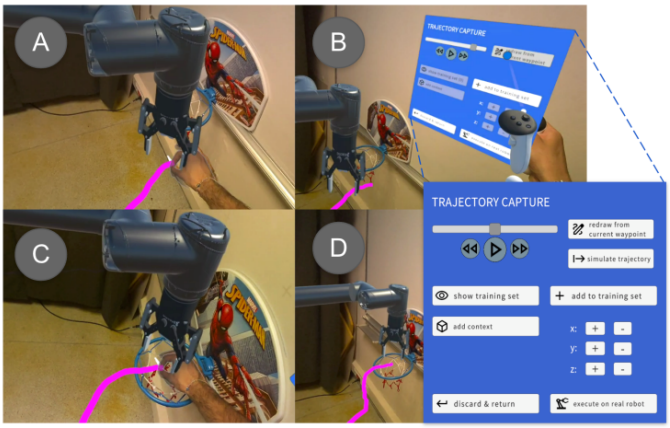

Fig. 3. The user (a) draws a trajectory to the hoop, (b) rewinds the trajectory and chooses to redraw it from the current way-point using the UI menu, (c) redraws the trajectory, and (d) finalizes the trajectory.

trajectory, they can redraw the trajectory from the current way-point. This interactive approach enables precise control of the robot's movements and ensures that the final trajectory is both accurate and safe for the desired task.

3) Real-Time Hand Following: The user may prefer having the simulated robot follow the position and orientation of their hand, as shown in Fig. 1. This enables the user to plan more effectively and refine trajectories in a safe and controlled environment, avoiding potential safety issues or collisions. The immediate feedback allows previewing how the robot will navigate through the real objects.

4) ML Training and Preview: The recorded trajectories can be either discarded or added to a training set, which can then be viewed or deleted, as shown in Fig. 4. RAMPA allows the user to add demonstrated trajectory to the training set after the simulation of the robot motion and/or physical robot execution. Moreover, the user can modify the recorded trajectory post-simulation and/or post-execution, which enhances the adaptability of the system and flexibility of the user during the demonstration collection process. The training set can then be used to train a model. To test the model, the user can add way-points to condition the trained model using virtual cubes,

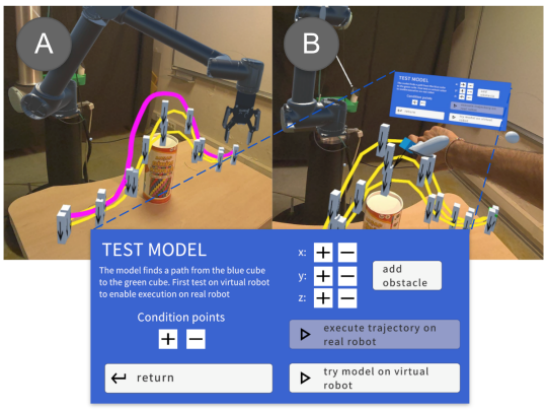

Fig. 4. Users can save and view trajectories they have recorded, then train an ML model, condition it using virtual cubes, and generate a new trajectory.

as demonstrated in Fig. 4. This process facilitates the tasks of training, conditioning, and sampling trajectories efficiently.

5) Execution on the Real Robot: The execution of the recorded trajectories and the sampled trajectories from the trained model can be performed using the virtual user interface. This capability actualizes the transfer of trained tasks from simulation to real-world execution.

B. Implementation

The implementation can be divided into four parts: In the game engine, where Unity is used, the main functionalities are implemented using *Meta Quest* API packages. The ROS part manages the interactions between the application and the IK service and the models used for training, along with handling the communication between the application and the actual robot. The actual robot is used to record the demonstrated trajectories and execute the skills effectively upon request. Finally, the XR headset, specifically *Quest 3*, deploys and operates *Unity* application. The following sections examine each part in detail, complemented by an illustration of the technical workflow in Fig. 5.

1) Unity: The simulated robot is integrated into *Unity*, using the *URDF Importer* package of *Unity*, where the simulated robot replicates the kinematics, dynamics, and physical properties according to the Unified Robot Description Format (URDF) description of the robot. A menu is also implemented, enabling the user to draw and adjust trajectories, switch between different trajectory capture modes, and perform the previously mentioned ML tasks.

Detecting the hand movements are done using *Meta Quest*'s hand gesture recognition and the related APIs. The user can choose from the UI menu provided before starting to demonstrate the trajectory whether the positions in the trajectory are recorded with a fixed orientation or if the orientation of the user's hand is also captured. The end-effector's position is determined to be the position of the user hand pinch. If the user chooses to capture orientation, the hand orientation information provided by the related *Meta Quest* API is used, and the palm-facing-down orientation of the hand is mapped to the orientation in which the end-effector's tool axis aligns with the negative vertical direction. The user can also choose whether the simulated robot replicates the trajectory after the demonstration or the robot follows the hand in real time.

To enable co-location of the simulated robot with the actual robot, the user places the left controller of the headset to a stable 3D-printed tool. After selecting auto-calibration in the UI, the simulated robot teleports to the position of the

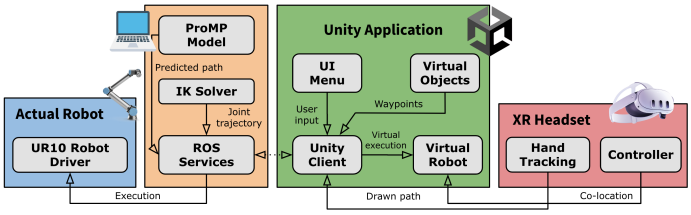

Fig. 5. Technical workflow of RAMPA.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

controller with a relative offset. For our case, we 3D-printed a tool that can stand on the base of *UR10*; however, it is important to note that any fixed, stable object that can hold the controller can be used to auto-calibrate, alongside necessary offset calculations.

If the user prefers real-time hand-following with the simulated robot, an off-the-shelf IK solver is continuously queried to compute the trajectory segment from the previous hand location to the current. This is done as the user continues to draw the trajectory as shown in Fig. 2. In this study, the TRAC solver [33] is used for IK computations. Obtained trajectory segments are simultaneously sent to *Unity* for illustration purposes via the simulated robot. If the user does not want to activate real-time hand-following, then the entire trajectory data, containing uniformly sampled and time-stamped 6D Cartesian positions, is recorded first and then sent to the ROS client at once. In either case, the trajectory sampling rate is fixed at 200 milliseconds. Immediately after receiving waypoints, the ROS client uses the IK solver to compute the necessary joint angles to execute the trajectory upon the user's request. During execution, a warning is displayed if the simulated robot collides with any object in the surroundings.

RAMPA enables users to replay entire trajectories or readjust portions of any length using the menu illustrated on the right-hand side of Fig. 3. It also allows saving trajectories and viewing waypoints, which are shown with yellow lines and black arrows in Fig. 4. One of the most crucial features that RAMPA provides is that the recorded data can be used directly to train ML-based models. The user can also condition the ML-based trajectory model by directly observing the actual obstacle and placing a virtual marker accordingly, as shown in Fig. 4. User-specified time-stamps and 6D Cartesian positions of the placed virtual markers are used to condition the model. The output of the conditioned model is then sent to Unity and simulated by the virtual robot for validation purposes.

2) ROS Client: The ROS server remains on standby to accommodate the requests from the *Unity* application. Upon initialization, the application establishes a TCP connection using the *ROS-TCP-Connector* package from *Unity*. Within the server, multiple ROS services are utilized to manage various types of operations initiated by the *Unity* application. These services operate within a single process and require unique message types that the *Unity* client complies with.

Upon the user's request, the ROS client reformats the recorded data and trains a new model from scratch. Since neural-network-based frameworks require long training procedures, we demonstrate this feature by training ProMPs.

3) Actual Robot: For the execution of the trajectory on the real robot, a TCP connection is set up between the server and the actual robot. By publishing states from the recorded trajectory in the virtual environment, the actual robot follows the published states.

4) XR Headset: The *Unity* application is deployed on the headset for complete operation with AR experience. For the co-location of the simulated robot with the actual robot, the left controller's position and rotation are used with relevant offset values, as illustrated in Fig. 2-A.

IV. EVALUATION

We conducted a comprehensive evaluation using both quantitative and qualitative metrics to assess the competence and performance of RAMPA and compare it with Kinesthetic Control (KC). These assessments include measuring the task-completion time, evaluating the performance of trained MP, the smoothness of trajectories, and user feedback concerning usability, user experience, and task load through questionnaires.

A total of 20 participants (18 male and 2 female, ages 20-36, M=23.5, SD=3.5) were recruited for the user study. Participants include 10 people with experience in robotics, where 7 of them additionally had experience in AR (experts), and 10 people with no prior experience in AR and robotics (novices). Permission for the user study is taken from the Bogazici University (No:2024/22).

We have conducted an experiment including three robotic tasks; both demonstrated using RAMPA system (RAMPA condition) and the physical robot in Kinesthetic Control (KC) condition. Participants were equipped with the *Quest 3* XR headset. We used the Universal Robot UR10, and the experiment was conducted in a robotics laboratory on campus.

A. Procedure

We arranged the user study so that half of the participants first used the RAMPA system and then moved the robot in KC mode, while other participants performed these tasks in the opposite order. Before participants started performing tasks, we offered an overview and explanation of the procedure and experiment. Specifically, we showed users how they can utilize trajectory playback, edit, save and deletion features of RAMPA, if desired or whenever necessary. Regarding KC mode, the users are informed that they will have to demonstrate from scratch, if they desire to modify their demonstrated trajectory. The users are also told that saving and replaying demonstrated trajectory in KC mode will be handled manually by the experimenter. After participants became familiar with using the RAMPA system and moving the real robot in KC mode, we asked them to perform **three main tasks**:

a) Warm-up Exercise: Users were asked to move and control the physical robot. To familiarize participants with the RAMPA system, they were asked to place a 3D printed controller attachment to ensure the real and virtual robots are correctly calibrated position-wise and draw a simple trajectory.

b) Basketball Task (Task 1): Participants were requested to draw a trajectory where when the simulated robot mimics the route drawn, it can pass a ball it holds through a basketball hoop. The robot initially holds a ball and releases it at the end of the route, which is executed externally by the experimenter.

c) Chain-Hook Task (Task 2): Inspired by crane operations for logistics in the industry, a chain hanging from the ceiling and an object on the table with an attached hook were prepared. Users were requested to draw a trajectory for the robot to perform a sequence of actions: moving the chain towards the object, connecting it with the hook, lifting the object from the table, and placing the object back on the table.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

d) Obstacle Avoidance Task (Task 3): Users were requested to train and test a model for an obstacle avoidance task. They were asked to draw trajectories by avoiding the total of 3 objects with changing sizes and train the model using these trajectories. Then, we placed an object of a different size from the previous objects and asked participants to test their model by specifying it via a point above the object.

We recorded the completion time for each task. In tasks 1 and 2, the duration for demonstrating a single successful robotic trajectory and the number of trials needed was recorded. In task 3, we measured the time required to demonstrate three distinct robot trajectories, and also saved the demonstrated trajectories for subsequent analysis, allowing for the comparison of their smoothness and similarity among trained MPs. For training, ProMP is used and the hyperparameter of weights per dimension is set to 20. Upon completion, participants were also asked to complete a survey. During the user study, demonstrations discarded due to technical issues (e.g., headset or robot malfunctions) were not considered.

B. Results & Discussion

a) Task Completion Time: We measured the task completion time to evaluate the efficiency of the RAMPA system compared to KC. Our analysis, as presented in Fig. 6, revealed that the average task completion time is shorter while participants utilize the RAMPA system. A paired t-test confirmed significantly shorter completion times in the RAMPA condition across all tasks: Task 1 (t = 11.83, p <0.001), Task 2 (t = 13.93, p <0.001), Task 3 (t = 7.55, p <0.001).

In the KC condition, Table II shows that the task completion time difference between expert and novice users was 22.6% for Task 1 and 34.5% for Task 2, indicating a notable difference due to the increased complexity of the demonstration for Task 2 as expected. On the other hand, when utilizing the RAMPA system, the task completion time difference was remarkably reduced. Our analysis thus suggests that RAMPA allows users to demonstrate robot trajectories efficiently and lessens the impact of prior experience on task performance.

b) Trained Movement Primitives: To compare the trajectory generation performances of the models, a ground truth trajectory is required for the new environment where these models are tested. For this, we initially trained a contextual ProMP model using the trajectories demonstrated during Task 3, using the heights of the objects as the model's parameter. We considered the trajectory sampled from this model as the ground-truth trajectory. Next, we computed the Mean Squared Error (MSE) between the ground-truth trajectory and the generated trajectory from the trained and conditioned ProMP by the user during task 3. The result shows that the mean of MSE values is 0.0017 (SD = 0.0024) for the KC and 0.0014 (SD = 0.0011) for RAMPA, and indicates that the ProMP model shows comparable performance when trained on trajectories derived using either RAMPA or KC.

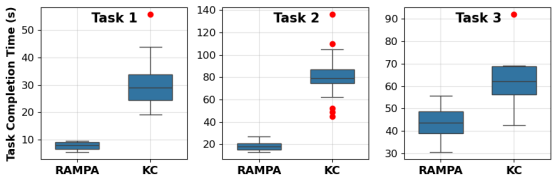

Fig. 6. Comparison of task completion times of RAMPA and Kinesthetic Control. The box shows the interquartile range (IQR), including the middle 50% of the data between the first quartile (Q1) and the third quartile (Q3). The whiskers extend from the edges of the box to 1.5 IQR of the Q1 and Q3, and the rest of the data points are considered outliers, shown as red points.

TABLE II
COMPARISON OF TASK COMPLETION TIMES

		RAMPA	KC
Task 1	Experts	**7.9s ± 1.3**	27.6s ± 5.2
	Novices	**7.5s ± 1.4**	33.3s ± 10.0
Task 2	Experts	**17.7s ± 3.4**	69.5s ± 15.6
	Novices	**19.2s ± 4.8**	92.1s ± 18.9
Task 3	Experts	**46.8s ± 7.0**	63.4s ± 5.3
	Novices	**40.9s ± 7.0**	60.7s ± 13.0

c) Trajectories Smoothness: We analyzed the smoothness of trajectories recorded using RAMPA and KC. The results reveal that the RAMPA demonstrations showed a lower jerk (0.004 m/s^3 ± 0.002) compared to the KC demonstrations (0.045 m/s^3 ± 0.011) while exhibiting comparable performance in both the deviation (RAMPA: 0.111 m ± 0.018, KC: 0.109 m ± 0.010) and the variation (RAMPA: 0.070 m^2 ± 0.021, KC: 0.073 m^2 ± 0.012), implying superior or comparable smoothness across all metrics. This analysis indicates that RAMPA facilitates efficient trajectory demonstration while maintaining the consistency and quality of demonstrations.

d) Number of Trials: All participants successfully completed tasks 1 and 2 in both conditions. For task 1, the number of trials required by each participant was 1, except for one novice user attempting twice while using the RAMPA system. For task 2, the average number of trials is 1.111 (SD = 0.314) for expert participants to succeed in both KC and RAMPA conditions. For novice users, the mean of the number of trials is 1.2 (SD = 0.4) in KC and 1.4 (SD = 0.49) in RAMPA, which is not significantly different (t = -1.5, p = 0.168). A slight increase in the number of trials for task 2 is expected, given the complexity of the demonstration compared to task 1. Our analysis suggests that the number of trials did not show a statistically significant difference between KC and RAMPA conditions for both experts and novices in tasks 1 and 2.

e) Survey Results: The survey consists of *System Usability Scale* (SUS) [5], the shorter version of the *User Experience Questionnaire* (UEQ) [6], the *NASA Task Load Index* (NASA-TLX) [7], and additional questions to compare the different aspects of RAMPA and KC. We utilized the SUS to evaluate and compare the usability of RAMPA and KC. We obtained a score of 82.75, which shows that the RAMPA system is within A grade (i.e., 90%-95%) while KC achieved a score of 62.38 which places KC within D grade [34].

We employed a subset of questions from the SUS to compare RAMPA and KC. As shown in Fig. 7 , the RAMPA system has superior scores across positive statements such as Q1, Q3, Q7, and Q9, which suggests that RAMPA provides an easy-to-use framework. The reported scores for Q4, Q7,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

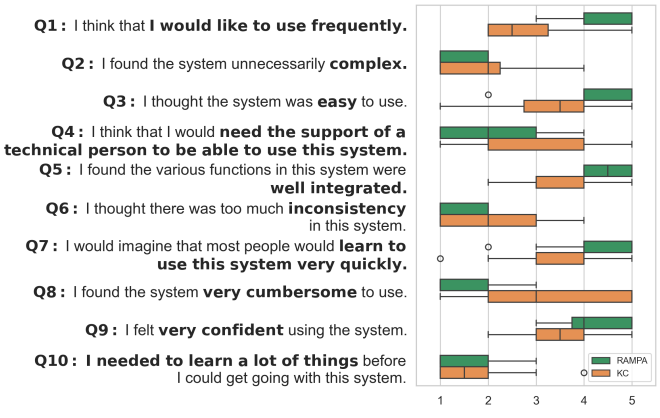

Fig. 7. The comparison of SUS results of KC and RAMPA conditions.

and Q8 suggest that manually moving the robot might be perceived as more demanding and complex compared to using RAMPA. These findings support that RAMPA enhances the user experience and satisfaction while simplifying the robotic trajectory demonstration process.

To evaluate the perceived safety of the RAMPA system, participants were asked to express which method they found safer. Reported responses were overwhelmingly positive towards RAMPA, with 95% of participants (19 out of 20) indicating that using RAMPA is safer than KC. This result shows increased perceived safety with RAMPA.

We employed the shorter version of the UEQ to further evaluate the user's experience of using the RAMPA system. As shown in Fig. 8, the RAMPA system has positive evaluations across both pragmatic and hedonic qualities. Particularly, RAMPA received a score of 2.09 on pragmatic quality, 1.94 on hedonic quality and 2.01 on overall scales, which puts the RAMPA system in the range of the top 10% results [6].

The analysis of the NASA-TLX survey with a paired t-test reveals that participants using RAMPA were more satisfied with their task performance ($t = 2.08$, $p = 0.044$) and felt less frustration ($t = -2.518$, $p = 0.016$) while indicating that RAMPA requires less physical demand ($t = -6.92$, $p < 0.001$) and effort ($t = -5.04$, $p < 0.001$) in Fig. 9. In addition, we used the RAW-TLX procedure [35] to compute aggregated workload scores. The resulting scores are 27.08 for RAMPA and 44.33 for KC, indicating that participants generally perceived a lower workload while using the RAMPA system compared to KC condition. The increased performance satisfaction with

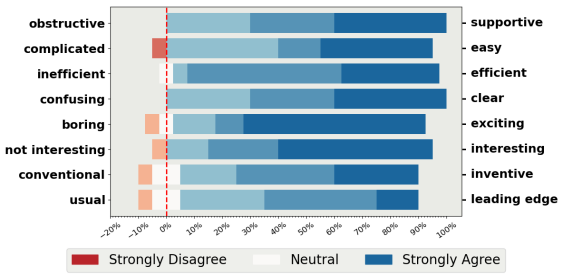

Fig. 8. UEQ results of the RAMPA condition.

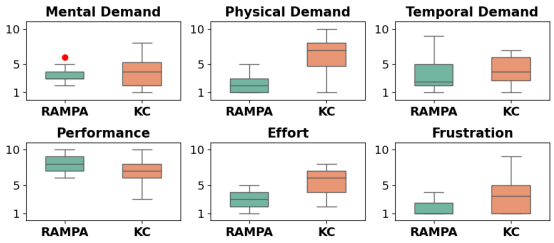

Fig. 9. Comparison of NASA-TLX results of the RAMPA and KC conditions.

reduced physical demand, effort, and frustration suggests that RAMPA provides a user-friendly interface to perform tasks effectively. These findings show that RAMPA enhances not only task efficiency but also user experience and comfort.

Participants were asked which approach they would prefer to use for demonstrating robotic trajectory. The reported answers show a significant preference towards RAMPA with 95% (19 out 20 participants), which, considered with previous evaluation results, indicates that RAMPA strongly facilitates the efficacy and accessibility of the robot programming process.

f) Qualitative Feedback: We collected qualitative user feedback from participants to provide a more comprehensive evaluation of RAMPA. One participant stated: *"The AR system made me feel safe throughout, and I didn't have to worry about damaging the robot"* (P20), highlighting the safety advantages of RAMPA. Regarding efficiency, one expert participant noted: *"Despite my experience in the domain, it is intimidating even for me to manually control the industrial robots.* RAMPA *simplifies the data collection part of LfD and speeds up my research"* (P11). The system's intuitiveness was highlighted: *"The controls with hand gestures being straightforward"* (P1) and *"AR integration is done seamlessly. It is easy to understand the trajectory of the robot"* (P19).

g) Auto-Calibration Accuracy: To assess the auto-calibration method, we conducted an experiment where we moved the robot to a pre-defined point on the table using the RAMPA system. Firstly, we established a reference point by moving the robot manually to that point and recording the end-effector position. Then, we performed auto-calibration, demonstrated a trajectory using the RAMPA system, and recorded the end-effector position, which was repeated 10 times. The resulting mean of the position error relative to the reference point is 0.0196 m (SD = 0.0134).

V. LIMITATIONS

Our system allows collision detection between the simulated robot and the scanned static environment represented with meshes. It currently does not support visual detection of dynamic objects [36], and interaction of the simulated robot with them. For tasks that require intense robot-object interactions, the dynamic objects should be automatically registered as interactable objects to the *Unity Engine* simulator on-the-fly, which requires realistic modeling of all physical properties such as mass, center of mass, detailed shape and density. While functional, auto-calibration might be time-consuming; easy-to-use solutions like trackable QR codes can facilitate

this process [37]. Moreover, controlling the gripper through the AR interface would improve the completeness of the system, diminishing the need for external operations. Also, a visualization of the robot's workspace can help users grasp its limitations, making the demonstration process less error-prone. RAMPA currently advances PbD research by replacing kinesthetic control, lacking a task-centered focus. Future extensions may incorporate egocentric camera views, computer vision, or semantic understanding to mimic daily user interactions and utilize real-world objects. Even though we use the *UR10* robotic platform and ProMPs for convenience, we designed RAMPA to be robot- and model-agnostic. The system can be adapted to other robotic hardware if their URDF description and its TRAC IK [33] configuration are provided, necessitating only minimal modifications in the *Unity* application and the ROS client. The system can also be extended by integrating other ML models into the ROS client, if the model can be trained real-time. In addition, although the *Unity* application depends on *Meta Quest* APIs, it can be adapted to any headset, given it has a corresponding XR library with hand recognition.

VI. CONCLUSION

We introduced an AR-based framework that provides a comprehensive and intuitive platform for demonstrating robotic skill trajectories, integrating trajectory visualizations and modifications, real-time hand mimicry, and training and testing ML models. These functionalities simplified the demonstration collection in PbD procedures and enhanced the human-robot interaction by leveraging AR. We presented quantitative evaluations, supporting our initial hypothesis of increased effectiveness without sacrificing user experience or task precision. RAMPA illustrates the potential to lead to more accessible and efficient robotic programming for both novices and experts.

REFERENCES

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[2] 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, 2009.

[3] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for enhanced human-robot interaction and robotic interfaces," in *CHI*, 2022, pp. 1–33.

[4] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *NIPS*, vol. 26, 2013.

[5] J. Brooke, "Sus: A quick and dirty usability scale," *Usability Evaluation in Industry*, 1996.

[6] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Design and evaluation of a short version of the user experience questionnaire (ueq-s)," *IJIMAI*, vol. 4, p. 103, 01 2017.

[7] S. Hart, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Human mental workload/Elsevier*, 1988.

[8] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *ISRR*, 2005, pp. 561–572.

[9] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *IJRR*, vol. 42, no. 13, pp. 1133–1184, 2023.

[10] E. Ugur and H. Girgin, "Compliant parametric dynamic movement primitives," *Robotica*, vol. 38, no. 3, pp. 457–474, 2020.

[11] E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard, "Learning nonlinear multivariate dynamics of motion in robotic manipulators," *IJRR*, vol. 30, no. 1, pp. 80–117, 2011.

[12] X. Yin and Q. Chen, "Learning nonlinear dynamical system for movement primitives," in *SMC*. IEEE, 2014, pp. 3761–3766.

[13] R. Pérez-Dattari and J. Kober, "Stable motion primitives via imitation and contrastive learning," *IEEE Transactions on Robotics*, 2023.

[14] D. Blessing, O. Celik, X. Jia, M. Reuss, M. Li, R. Lioutikov, and G. Neumann, "Information maximizing curriculum: A curriculum-based approach for learning versatile skills," *NeurIPS*, vol. 36, 2024.

[15] Y. Yildirim and E. Ugur, "Conditional neural expert processes for learning movement primitives from demonstration," *RA-L*, 2024.

[16] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Auton Robots*, vol. 42, p. 529, 2018.

[17] C. Paxton, A. Hundt, Jonathan, Guerin, and Hager, "Costar: Instructing collaborative robots with behavior trees and vision," in *ICRA*, 2017.

[18] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama, "Robot programming by demonstration with interactive action visualizations." in *Robotics: science and systems*, 2014, pp. 1–9.

[19] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," *arXiv preprint arXiv:2402.05741*, 2024.

[20] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design," *Intl. journal of advanced robotic systems*, vol. 5, p. 1, 2008.

[21] M. Walker, T. Phung, T. Chakraborti, T. Williams, and D. Szafir, "Virtual, augmented, and mixed reality for human-robot interaction: A survey and virtual design element taxonomy," *Trans. on Human-Robot Interaction*, vol. 12, no. 4, pp. 1–39, 2023.

[22] X. Jiang, P. Mattes, X. Jia, N. Schreiber, G. Neumann, and R. Lioutikov, "A comprehensive user study on augmented reality-based data collection interfaces for robot learning," in *HRI*, 2024.

[23] Y. Cao, T. Wang, X. Qian, P. S. Rao, M. Wadhawan, K. Huo, and K. Ramani, "Ghostar: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality," in *Symposium on User Interface Software and Technology*, 2019, pp. 521–534.

[24] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *IROS*. IEEE, 2018, pp. 1838–1844.

[25] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *HRI*, 2018, pp. 316–324.

[26] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating robot arm motion intent through mixed reality head-mounted displays," in *ISRR*. Springer, 2020, pp. 301–316.

[27] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto, "Holo-dex: Teaching dexterity with immersive mixed reality," in *ICRA*, 2023.

[28] M. B. Luebbers, C. Brooks, C. L. Mueller, D. Szafir, and B. Hayes, "Arc-lfd: Using augmented reality for interactive long-term robot skill maintenance via constrained learning from demonstration," in *ICRA*, 2021, pp. 3794–3800.

[29] M. Diehl, A. Plopski, H. Kato, and K. Ramirez-Amaro, "Augmented reality interface to verify robot learning," in *RO-MAN*, 2020, p. 378.

[30] K. Lotsaris, C. Gkournelos, N. Fousekis, N. Kousi, and S. Makris, "Ar based robot programming using teaching by demonstration techniques," *Procedia CIRP*, vol. 97, pp. 459–463, 01 2021.

[31] J. Chen, D. Salas, M. Bilal, Q. Zhou, and W. Johal, "Mr. lfd: A mixed reality interface for robot learning from demonstration," 2024.

[32] S. Chen, C. Wang, K. Nguyen, L. Fei-Fei, and C. K. Liu, "Arcap: Collecting high-quality human demonstrations for robot learning with augmented reality feedback," 2024.

[33] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *Humanoids*, 2015, p. 928.

[34] J. Sauro and J. R. Lewis, *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[35] K. Virtanen, H. Mansikka, H. Kontio, and D. Harris, "Weight watchers: Nasa-tlx weights revisited," *TIES*, vol. 23, 11 2021.

[36] M. D. Dogan, E. J. Gonzalez, K. Ahuja, R. Du, A. Colaço, J. Lee, M. Gonzalez-Franco, and D. Kim, "Augmented object intelligence with xr-objects," in *ACM UIST*, 2024.

[37] M. D. Dogan, A. Taka, Y. Zhu, A. Kumar, A. Gupta, and S. Mueller, "Infraredtags: Embedding invisible ar markers and barcodes using low-cost, infrared-based 3d printing and imaging tools," in *ACM CHI*, 2022.