2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Human-robot interaction for robotic manipulator programming in Mixed Reality

Mikhail Ostanin, Stanislav Mikhel, Alexey Evlampiev, Valeria Skvortsova, Alexandr Klimchik, IEEE Member

Abstract— **The paper presents an approach for interactive programming of the robotic manipulator using mixed reality. The developed system is based on the HoloLens glasses connected through Robotic Operation System to Unity engine and robotic manipulators. The system gives a possibility to recognize the real robot location by the point cloud analysis, to use virtual markers and menus for the task creation, to generate a trajectory for execution in the simulator or on the real manipulator. It also provides the possibility of scaling virtual and real worlds for more accurate planning. The proposed framework has been tested on pick-and-place and contact operations execution by UR10e and KUKA iiwa robots.**

I. INTRODUCTION

Transition to Industry 4.0 paradigms pushes industries and businesses to utilize robotic systems more extensively, which results in their stronger engagement in active human-robot cooperation/coexistence. This transition increases productivity and improves the overall manufacturing process, eliminating human participation in dangerous operations. Usually, deployment of the general-purpose robotic manipulator(s) requires them to be explicitly programmed for a specific task. There are three main ways to program a robot: Online that made by human, Offline programming and Programming by Demonstration.

In online programming, a human operator changes robot's end-effector using an operator console. To program the robot, the operator moves it from point to point and saves each position individually. When the whole task is completed, the robot can execute the program, running from point-to-point. In offline robot programming, the control program is created in a specific programming software or/and simulator and then uploaded into the robot controller for execution. This approach generally allows the user to create easily scalable solutions for different robotic sells and optimize robot motions.

Friedrich et al. [1] presented an approach of teaching machines to complete a task without getting into writing complex lines of code known as "Programming By Demonstration" (PbD). PbD can be defined as teaching (programming) robots by manually demonstrating task(s) and making the robots learn from actions. PbD allows an operator to perform this action without technical knowledge about how to program manipulators, so less qualified staff is required for this task, and that can reduce payroll. The main disadvantage of this approach is the need for special equipment and / or support of teaching mode in the robot, which leads to the need to stop the technological process during the programming phase.

Further advancement of PbD can be achieved by Mixed Reality (MR) facilities. Milgram and Kishino [2] introduced MR as a combination of Augmented Reality (AR) and Virtual Reality (VR). MR is a technology where both real and digital worlds co-exist and interact in real time. This concept allows for extending the idea of PbD moving programming phase in the virtual world. MR allows not only to teach the robot, but also to test the program safety by a digital twin in the virtual environment before implementing it in the real world by the physical robot. MR also opens new options for faster prototyping, including simulation of the work cell, examine overall performance and efficiency of the robotics system with different setups. Visualization of sensory data and robot configuration is an essential benefit of MR, which gives it the ability to interact with the spatial environment using gestures, gaze, sound, and motion controllers.

Thus, PbD and MR combined open new possibilities for robotic manipulator programming and may facilitate development of HRI and extensive utilization of robots in various operations.

II. BACKGROUND

Interfaces on augmented, mixed, and virtual reality can enrich the Human-robot interaction (HRI) process and solve several critical problems, related to security, efficiency, and simplicity [3].

There are several AR-based interfaces for interacting with robots. Fang et al. [4] developed an AR-based robot programming system in which they copied a virtual model of a robotic arm and used a cube marker with six flat markers as a tool for human interaction with a virtual robot. Wassermann et al. [5] created a cloud-based system where they used the point cloud to filter the objects on a table. Later they applied AR to generate bounding boxes around those objects. To check collision detection, the authors filled the working space of a robotic manipulator model (KUKA kr6) with a virtual/augmented 3D grid. The boxes of this grid would turn red in case of any collision. In previous works, the AR scene

The work presented in this paper was supported by the grant of the Russian Science Foundation 17-19-01740.

Mikhail Ostanin, Stanislav Mikhel, Alexey Evlampiev, Valeria Skvortsova, Alexandr Klimchik are with the Center for Technologies in Robotics and Mechatronics Components, Innopolis University, Innopolis, Russia (emails : {m.ostanin, s.mikhel, a.evlampiev, a.klimchik}@innopolis.ru, v.skvortsova@innopolis.university)

is visualized through a desktop monitor. One example of AR mobile interface presents approaches for robot control of multi-robot systems in shared spaces with human operators [6]. VR devices are also used in research aimed at improving HRI. In particular, there are works devoted to remote control of robots using VR glasses [7], [8].

In 2015, Hoenig et al. [9] defined several benefits of using MR in various robotic applications. The authors explained that "spatial flexibility" (i.e., merging of virtual and physical worlds) could help the researchers to scale the environment and the number of robots in the virtual world. Testing the robots in the virtual world also eliminates any threat of harm. More robots can be added, and the swarms can be scaled, and MR makes debugging easier too. These effects, as described in their paper, revealed during several experiments comparing quad-copters performance in the physical and virtual world.

Recent advancements in electronics, optics, and graphics have made commercially available MR glasses accessible to the consumers. Microsoft HoloLens was the first MR holographic device. Several researchers evaluated the applicability of such devices in HRI domain. [10], [11] compared and checked Rethink Baxter's Robotic arm in different scenarios including MR, motion controller, and 2D visualization. Rosen et al. found out that MR head-mounted display improved overall process time by 62 percent, precision by 11 percent, and collision prediction accuracy by 16 percent. Guhl et al. [12] came up with a distributed system to program a robot intuitively using HoloLens. They replaced the real robot model with a virtual one (a marker determined the robot's origin position). Quintero et al. [13] presented an AR robotic system for two trajectory types interactions: free space trajectory and contact surface trajectory. They found that AR robotic interface gives better performance than kinesthetic teaching.

In our previous work [14], we proposed a methodology and an interface for interactive robot programming using MR. We used Microsoft HoloLens and two robots (KUKA iiwa and Agilus) as a part of our implementation. MR system based on Unity3D and Mixed Reality Toolkit-Unity. Our implementation of the system could be broken down into four elements: Application Manager, Geometrical Path Planner, Trajectory Planner, Simulator. Each part was implemented on the HoloLens application side. Robot programming was done in a step sequence. Firstly, the user built a geometric path that was based on points placement and connects them in different ways (Point to point, Line, Arc, User path). Secondly, it was trajectory planning for the chosen robot (iiwa or Agilus). Thirdly, the user could run a simulation. Finally, the trajectory was executed on the robot.

The new system that we developed in our current research effort has some differences and improvements compared with the previous work. We tested the system on two robots: KUKA iiwa and Universal Robot UR10e. In the previous work, the system architecture modified all computational tasks which were concentrated on the side of HoloLens. In the current work the main computational tasks are executed on the Robot Operating System (ROS) side. Besides, the visual component of path planning was changed, and interaction has becomes more intuitive. The functionality of the system was also expanded with a new type of the robot state visualization, the ability to add various tools and grips, the ability to visualize the working space of the robot. Also, there were new unique possibilities based on MR: automatic positioning of the robot in the space, building a path around obstacles and the ability to scale the path together with real space. Overall, the new system has a more intuitive interface and broader functionality for programming industrial robots in MR.

III. SYSTEM COMPONENTS

The developed system component is shown in Fig. 1. It can be divided into three main parts: visualization and interface on Microsoft HoloLens, the computing part on ROS Kinetic and robots UR10e and KUKA iiwa LBR 14 as a real-world part. HoloLens is MR glasses that we used for interaction, user action recognition, and translation them to ROS. ROS Kinetic provides communication between system components. Robotic manipulators execute the user commands and send the state information to ROS.

We used the Unity3D game engine and Mixed Reality Toolkit for HoloLens development. Our application has an interface, geometrical path planning, spatial mapping, and virtual models. In displayed block diagram in Fig. 1 arrows show connections between main blocks. Robots have access to different tools and grippers. Spatial Mapping block is responsible for visualizing the real space in which the user is located. The block of geometric path planning has the primitives for path visualization and scrips for automation path settings which works with cooperation with Spatial Mapping and robots models.

The obtained robot path is published to the ROS. The function of the trajectory planner is to define such time sequence of velocities and accelerations that the trajectory is feasible. In this project, we use "MoveIt" framework inside that block. The obtained results are published to the robot controllers and/or to the virtual robot simulator. Another ROS module is Robot Configuration which returns the manipulator pose in space and robot joints state.

Each robot has API to ROS connection, for UR10e it is based on URScript, and for iiwa it is based on KUKA Sunrise. The trajectory planner message contains joint positions for each moment. Thus, the controller task is to connect them with the help of point-to-point (PTP) motion.

IV. IMPLEMENTATION AND INTERACTION

A. Basic functionality

This part of the article presents the basic functionality of the system, which is necessary for industrial robot programming. Fig. 2 gives examples of main virtual objects in our system.

Menu. To interact with the objects in MR world, a user can refer to the main menu to chose a suitable option and actions. It is shown in Fig. 2. The menu is responsible for

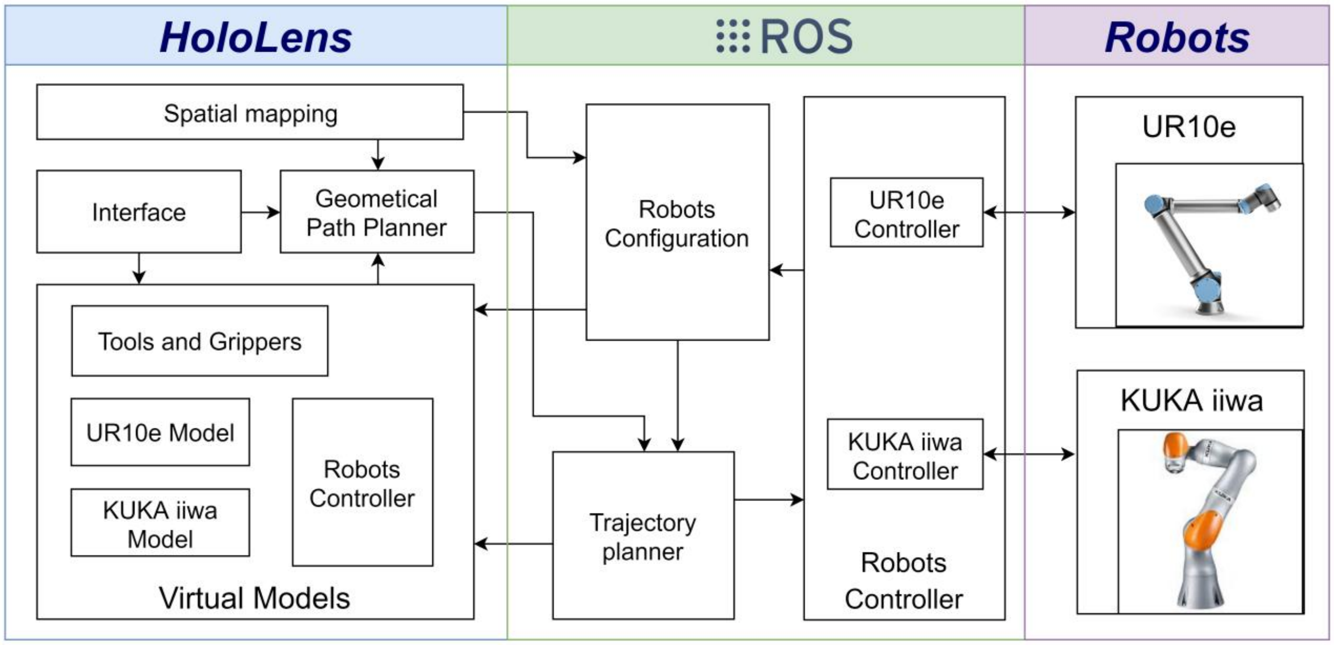


Fig. 1: Mixed reality based robot programming system components

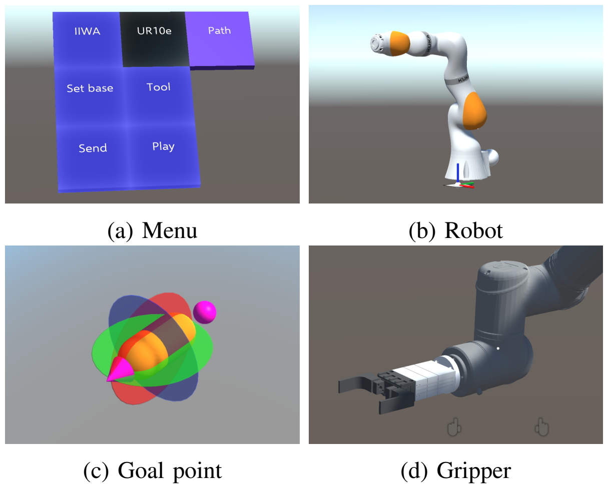


Fig. 2: Main virtual object of the system

access to different application parts, for robot settings and contains commands for working with the trajectory.

Robot state. Our interface can visualize the state of the robot. For the demonstration, we identified two types of information: the joint position and the pose of the end-effector.

Geometrical path planning. To get the trajectory of the robot, you first need to specify the geometric path of the end-effector motion. This path can be represented by a sequence of control points, each of them describes the position and orientation of the final link in the space - six degrees of freedom. A special hologram displays each control point. This hologram is shown in Fig. 2. It consists of a central part, by clicking on which you can move a point in space. It also includes circles, by clicking on which and moving to the right and left, you can set the orientation around the normal to this circle. Another way to set a control point is to place it on the cursor, which indicates a point on the object's surface with orientation normal to this surface. Thus, the user indicates the position and orientation around the normal, and the remaining 2 degrees of freedom are determined automatically.

The control points can be connected in one of various ways: PTP, line, and arc. Each option corresponds to the standard commands presented in the robot controllers and is selected in the trajectory settings menu. An additional way to connect the points is to draw a path by hand. By this feature, user can build more complex trajectory quickly.

For the control points, we can specify not only position and orientation but also gripper actions, such as opening and closing. The system has a small library with tools and grippers, each of that has its dimensions, which are taken into account for path constructing.

Trajectory generation. If the system works with different robots, it requires an efficient solver for the inverse kinematics (IK) problem. In many cases, it is possible to find solution analytically, in particular, when the robot is serial and has six joints, 3 for position and 3 for orientation. A unified framework can be found, for example, in [15]. Nevertheless, an industrial robot can have a different number of joints or kinematic structure, and it will be challenging to find the analytic solution. The numerical solution of the IK problem is a universal approach and can be applied to different robot constructions. When the initial position and the Jacobian are known, robot configuration in the desirable Cartesian state can be found with the help of numerical integration. In our work, we used the "MoveIt" framework for solving the IK problem and for trajectory generation, which is a part of ROS environment. This tool builds a robotic manipulator model using Unified Robot Description Format (URDF) files. It takes into account joint limits, collisions, and singularities.

The trajectory planers were implemented as ROS nodes. User command includes the name of the robot, gripper state, type of point connection, the junction and the sequence of desirable positions and orientations in Cartesian space. When the planer gets a new message, it reads the chain of command points and connects them with the desired type of line to get the path. Then the planner calculates accelerations and velocities that should be applied to get the fastest possible motion along the path. The obtained results are converted into the joint space. If the trajectory is feasible, the determined values are published for further application on the real robot or its simulator.

Simulator. The simulator is responsible for displaying and reproducing the robot's trajectory obtained during planning. It allows the user to assess the correctness of the movement of the robot and to verify the created program. This module reduces the number of errors and preserves the integrity of the real robot. MR provides an important feature for the simulator, namely, collision detection of a virtual model of robots with real space.

Workspace visualization. One of the most useful features of the system is that the workspace of a robotic manipulator can be visualized using HoloLens. It has a number of benefits: the system enables the user instantly assess the achievability of the points, alarming the user about what is in the working space of the robots and what is not. In addition to visual human control, the system can stop the robot or slow down the robot, if the operator is near the robot. Fig. 3 shows an example for UR10e.

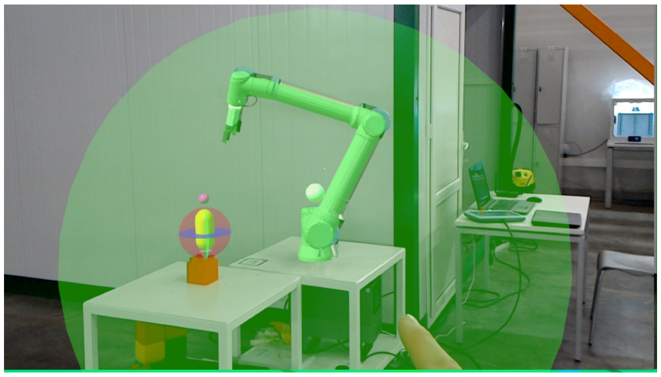


Fig. 3: Workspace visualization

B. Mixed Reality Features

In addition to the basic functionality, we have developed unique features that can be performed only in systems based on MR. The key aspect is the interlacing of the virtual and real environment. The first is the automatic positioning of the virtual robot model in space. The second is the construction of the shortest path with the bypass of obstacles. The third is scaling the path together with a part of the real space. And the fourth feature is a drawing of the path. Let's look at them in more detail.

1) Robot placement: The first part of the robot configuration is to find the real manipulator pose. It is extremely important because that has main influence to the point setting accuracy. As input, the algorithm takes a point cloud of the scene from MR glasses and a point cloud of the robot model. The algorithm consists of the following steps: preprocessing point clouds, clustering objects, searching for robot-like objects, finding the position of the robot. The algorithm works correctly for sparse and dense point clouds. The result of the algorithm is presented on Fig. 4

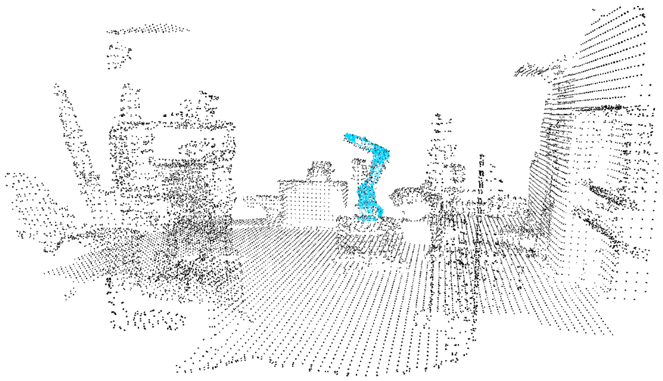


Fig. 4: Robot Recognition algorithm with sparse point clouds

Preprocessing scene and model point clouds is different since they have different origins. In the model point cloud, we remove the internal points by constructing a convex hull. For this we use an algorithm Jarvis algorithm [16]. In the point cloud of the scene, we remove the floor and ceiling using the random sample consensus (RANSAC) algorithm [17] for initialize the planes. It is important because we remove the connections between objects in scene if the form of floor. The next step is clusterization. We define the objects by Density-based spatial clustering of applications with noise (DBSCAN) algorithm [18]. It is a density-based clustering algorithm. It groups points that are close to each other. We initialize places of points accumulation as objects in the scene. After these operations, the algorithm calculates the volume of each initialized object in the scene and compares it with the volume of the model. Thus, the algorithm selects the objects most similar to the model. Localization of the robot is done through the iterative closest point (ICP) algorithm [19]. This algorithm iteratively searches for the smallest distance between the object in the scene and the model. We run this algorithm on all the objects most similar to the model in the scene and look for the minimum error of coincidence of point clouds. Select the transformation at which the error was the smallest.

The average running time of the algorithm is 23.1 seconds. Test point clouds of the scene with a known robot position were generated to verify the accuracy of the algorithm. The algorithm showed a positioning error of 9 mm in position and 3.1 degrees in orientation. The algorithm may produce an error if there are two similar-sized robots in the scene in

Algorithm 1 Robot localization in 3D point-cloud

```
Require: $Model$ - point cloud of configured robot
    $Scene$ - point cloud from HoloLens
Ensure: $HT$ - transformation from $Model$ to $Scene$ origin
 1: $Model, Scene \leftarrow Pre-processing(Model, Scene)$
 2: $Scene\_obj \leftarrow DBSCAN\_Clasterization(Scene)$
 3: $Model\_volume \leftarrow CalculationVolume(Model)$
 4: for $Object$ in $Scene\_obj$ do
 5:     $Volume \leftarrow CalculationVolume(Object)$
 6:     if $Volume \approx ModelVolume$ then
 7:         $ModelTransform \leftarrow$
                $FindModelInitialPosition(Model, Object)$
 8:         $HT\_obj$, $Accuracy \leftarrow$
                $ICP(ModelTransform, Object)$
 9:         if $Accuracy > BestAccuracy$ then
10:             $BestAccuracy \leftarrow Accuracy$
11:             $HT \leftarrow HT\_obj$
12:         end if
13:     end if
14: end for
15: return $HT$
```

the same configuration. In some cases, the algorithm gave an error at the stage of clustering objects. If this error occurs, the user is allowed to set the initial position of the robot, and then run the ICP algorithm only relative to the point cloud around the user-defined point.

2) Shortest path avoiding obstacles: The workspace of the manipulator can contain different obstacles that limit the robot movements. Most straightforwardly, to avoid obstacles the user should specify trajectory manually. Our solution provides a better user experience by performing automatic path planning with avoiding obstacles. It is possible to automatically produce an optimal path as HoloLens builds a map of the environment that can be exploited. To begin with, the user specifies the initial sequence of control points through the HoloLens interface. There are special connections between control points: the shortest path in Cartesian space and the shortest path in Joints space. Additionally, Hololens used to get spatial mapping. It contains mesh colliders that represent object surfaces. To check for collisions with these colliders Unity engine API is used.

Our obstacle avoidance algorithm for joints space is based on Rapidly-exploring Random Tree (RRT) [20]. RRT explores space in the form of the tree from the initial state of the manipulator. It returns the sequence of robot joint positions. Following these positions would be safe and roughly optimal in terms of distance. Average time of searching path is less than a second while performing multiple hundreds of iterations. In each iteration new node in the tree is added.

Another obstacle avoidance algorithm based on A*, it finds the shortest part in Cartesian space. For each robot, model workspace is known, therefore, in the beginning, it is divided into cells called voxels. Voxel representations for motion planning examined in [21]. A cell is the smallest volume of environment that can be considered as free or occupied. It should be small enough to have a good resolution of the workspace. We used a size of 4 cm. Each cell checked by Unity tools for obstacles, results collected into a boolean array. This array is a 3D occupancy grid that is updated before each run. Finally, A* algorithm runs.

Both obstacle avoidance techniques based on RRT or voxel representations of the environment are applicable for use in mixed reality. However, the first one checks links collisions continuously but the second one independent from the robot model. Using obstacle avoidance feature is simple, and it decreases the number of actions needed to create the trajectory. The experiment result shows in Fig. 5.

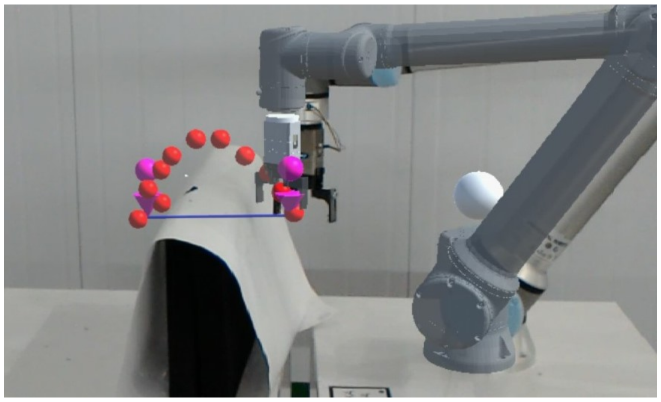


Fig. 5: Path with obstacle avoidance

3) Path scaling: Scaling is important for creating a more accurate path. Since the holograms are quite big, the accuracy of path settings without scaling is approximately 1-2 cm. By doubling the scale, we can increase the accuracy to about 0.5 cm. Fig. 6 shows an example. Interaction is based on two virtual cubes. First one determines the scalable area, and the second one demonstrates a scaled copy of virtual objects inside the small cube. It can be part of the path and the part of the Spatial mapping.

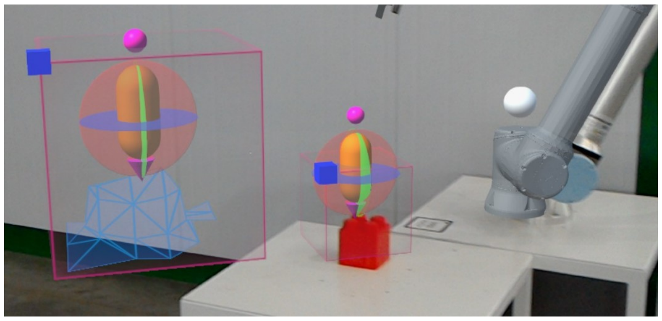
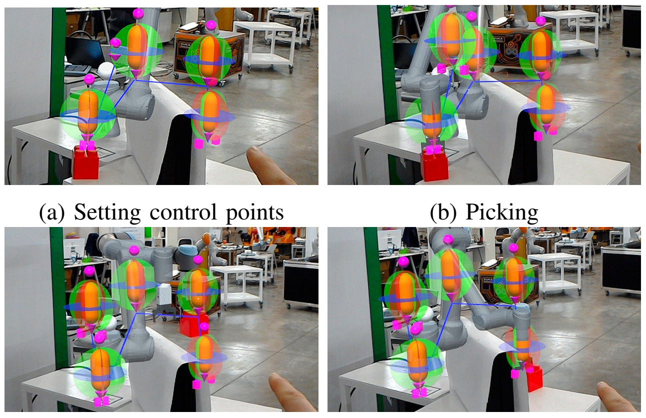


Fig. 6: Path scaling

V. Demonstration and Dissuasion

1) Pick and Place: The goal is to pick the object and move it to another place. The complexity of the solution depends on the types of objects and gripper. Soft gripper simplifies the problem, but when it is rigid, the orientation of the tool and value of impact should be taken into account, especially if the object is fragile.

The ability to see the object and interact with the marker allows choosing the position and orientation of the gripper in an optimal way. Firstly, the user should set up the start and the final position. Then he can add new via points; another variant is the program finds the shortest path automatically taking into account possible collisions. After getting the trajectory, robot motion can be visualized to check the solution correctness. In case of success, execution with a real robot could be done. The main steps of the process are shown in Fig. 7.


(a) Setting control points (b) Picking

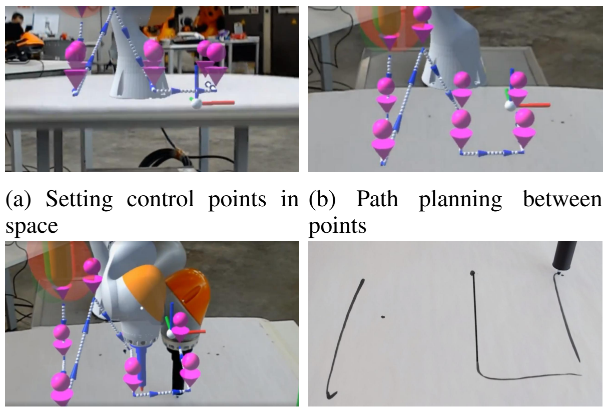
(c) Execution (d) Placement

Fig. 7: The basics steps of draw words by Industrial robot programming using mixed reality

2) Contact operation: Another typical problem for manipulators is to perform a contact operation while the tool moves along the predefined path. It can be, for example, welding, gluing, painting, and so on. In our experiment, we were modeling it with writing the text. In order to perform this task, the system should not only be able to send trajectory points to the robot but also to correctly recognize the location of the robot and surface.

Our system functionality allows the user to set the points precisely to the real object surface. Firstly, the user should arrange geometrical sequence points of the drawing contour. After that, the coordinates of these points take ROS for trajectory planning. The result of this step will be sequence joints states. The user can ensure that the robot actions are correct by simulating the movements of the robot according to the joint trajectory. In Fig.8 you can see more about necessary steps of a draw by industrial manipulator and result of drawing abbreviation ”IU”. The force is not controlled in our system, for perform contact operation we used the specific tool with spring inside.

Thus, robotic manipulator programming using MR interface with conventional programming through tablet or kinesthetic programming gives essential benefits in terms of time to write a program [10], [13] reduces the number of errors due to virtual simulation in virtual environment during the debug. It should be stressed that learning how to


(a) Setting control points in space (b) Path planning between points

(c) Simulation and execution of drawing (d) Results

Fig. 8: The basics steps of draw words by Industrial robot programming using mixed reality

use MR interfaces takes less time compared to a classic 72-hour training course for industrial robots programming [10]. The ability to scale the path and utilization of additional cameras/sensors reduces the impact of MR interfaces on the robot positional accuracy and in some cases may be used for quality estimation after the technological operation. In addition to programming robots, such systems can be used in production planning. Due to the visualization of the workspace and the possibility of substituting any robot for virtual task execution, it is possible to adjust the production line layout and to locate the tasks in the optimal zones.

VI. CONCLUSIONS

In this paper, we presented an MR-based framework for robotic manipulator programming. HoloLens glasses were used for operator gesture recognition and object visualization.The framework was tested with industrial collaborative manipulators KUKA iiwa 14 and UR10e. ROS was communication between the system components. User can specify the geometric path of motion with the help of control points, each of which describes the position and orientation of the end-effector. The obtained path is transformed into a joint space trajectory for the desired robot and can be used either in the simulator or in the real manipulator. The key and unique features of developed MR interactive programming system are (i) robot placement based on Point Cloud analyses, (ii) obstacle avoidance based on the graph description of the free space in the robot area, (iii) scale feature for a more accurate path planning. The proposed framework has been tested with such typical problems as pick-and-place and performing a contact operation.

In the future, we will provide a user study to compare our MR interface with 2D and kinetostatic interfaces. We are planning an extension of MR-based programming system via integration multi-user capabilities, automatic work-cell reconstruction in the virtual environment, teleportation facilities for remote objects as well as integration AR and VR devices in MR-based system.

References

[1] H. Friedrich, R. Dillmann, and O. Rogalla, “Interactive robot programming based on human demonstration and advice,” in *Sensor Based Intelligent Robots*, Springer, 1999, pp. 96–119.

[2] P. Milgram and F. Kishino, “A taxonomy of mixed reality visual displays,” *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[3] T. Williams, D. Szafir, T. Chakraborti, and H. Ben Amor, “Virtual, augmented, and mixed reality for human-robot interaction,” in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2018, pp. 403–404.

[4] H. Fang, S. K. Ong, and A. Y.-C. Nee, “Robot programming using augmented reality,” in *2009 International Conference on CyberWorlds*, IEEE, 2009, pp. 13–20.

[5] J. Wassermann, A. Vick, and J. Krüger, “Intuitive robot programming through environment perception, augmented reality simulation and automated program verification,” *Procedia CIRP*, vol. 76, pp. 161–166, 2018.

[6] J. Lambrecht and J. Krüger, “Spatial programming for industrial robots: Efficient, effective and user-optimised through natural communication and augmented reality,” in *Advanced Materials Research*, Trans Tech Publ, vol. 1018, 2014, pp. 39–46.

[7] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, “Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–9.

[8] J. I. Lipton, A. J. Fay, and D. Rus, “Baxter’s homunculus: Virtual reality spaces for teleoperation in manufacturing,” *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 179–186, 2017.

[9] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, “Mixed reality for robotics,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 5382–5387.

[10] 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,” 2017.

[11] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, “End-user robot programming using mixed reality,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2707–2713.

[12] J. Guhl, S. Tung, and J. Kruger, “Concept and architecture for programming industrial robots using augmented reality with mobile devices like microsoft hololens,” in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2017, pp. 1–4.

[13] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1838–1844.

[14] M. Ostanin and A. Klimchik, “Interactive robot programing using mixed reality,” *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 50–55, 2018.

[15] M. Brandstötter, A. Angerer, and M. Hofbaur, “An analytical solution of the inverse kinematics problem of industrial serial manipulators with an ortho-parallel basis and a spherical wrist,” in *Proceedings of the Austrian Robotics Workshop*, 2014, pp. 7–11.

[16] M. I. S. Franco P. Preparata, *Computational Geometry Algorithms and Applications Third Edition*. 2008, p. 388.

[17] R. C. B. Martin A. Fischler, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Comm. ACM*, pp. 381–395, 1981.

[18] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “Dbscan revisited, revisited: Why and how you should (still) use dbscan,” *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, p. 19, 2017.

[19] F. Pomerleau, F. Colas, and R. Siegwart, “A Review of Point Cloud Registration Algorithms for Mobile Robotics,” *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[20] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[21] A. Nakhaei and F. Lamiraux, “Motion planning for humanoid robots in environments modeled by vision,” in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, Dec. 2008, pp. 197–204.