Contents lists available at ScienceDirect

Robotics and Computer Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length Article

Augmented reality-assisted robot programming system for industrial applications

S.K. Ong[*], A.W.W. Yew, N.K. Thanigaivel, A.Y.C. Nee

Mechanical Engineering Department, Faculty of Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576, Singapore

ARTICLE INFO

Keywords:
Augmented reality
Robot programming
Human-robot interaction

ABSTRACT

Robots are important in high-mix low-volume manufacturing because of their versatility and repeatability in performing manufacturing tasks. However, robots have not been widely used due to cumbersome programming effort and lack of operator skill. One significant factor prohibiting the widespread application of robots by small and medium enterprises (SMEs) is the high cost and necessary skill of programming and re-programming robots to perform diverse tasks. This paper discusses an Augmented Reality (AR) assisted robot programming system (ARRPS) that provides faster and more intuitive robot programming than conventional techniques. ARRPS is designed to allow users with little robot programming knowledge to program tasks for a serial robot. The system transforms the work cell of a serial industrial robot into an AR environment. With an AR user interface and a handheld pointer for interaction, users are free to move around the work cell to define 3D points and paths for the real robot to follow. Sensor data and algorithms are used for robot motion planning, collision detection and plan validation. The proposed approach enables fast and intuitive robotic path and task programming, and allows users to focus only on the definition of tasks. The implementation of this AR-assisted robot system is presented, and specific methods to enhance the performance of the users in carrying out robot programming using this system are highlighted.

1. Introduction

Industrial robots are important components in manufacturing systems, and have been applied in a wide variety of applications, such as welding, material handling, assembly and painting. Robots are invaluable in manufacturing environments due to their high reliability, precision, repeatability and predictability. Once the robots have been programmed correctly to perform certain manufacturing operations, these processes can be achieved with high speed and precision. In particular, the application of robots in standard manufacturing tasks, such as materials handling, spot welding, and spray painting, can achieve high productivity and efficiency [1].

There are two main robot programming methods, namely, online programming methods, such as the use of a teach pendant and the lead-through method, and offline programming methods [2]. Despite reducing the need for writing programming code manually, current conventional robot programming methods are still time-consuming. As a result, the robot programming process is costly and time-consuming for programming complicated tasks. Robots have to be re-programmed frequently for high-mix low-volume products; thus, the cost of employing robots is prohibitively high for the small and medium enterprises (SMEs).

The most widely-used programming method for industrial robots is the use of a teach pendant. It is a handheld device that allows users to move a robot with a joystick to record the path traversed by the robot. One difficulty of using the teach pendant is that a robot usually has more degrees of freedom than the joystick, such that the programming of the rotation and translation of the robot end-effector is typically performed separately. A user can define a Cartesian coordinate system that is more intuitive, e.g., by aligning the axes of the coordinate system to the edges of a table or a specific workpiece. However, the alignment of the motion of the robot along the Cartesian coordinate system has to be adjusted when the perspective of the user changes.

In the lead-through programming approach, a robot manipulator is held by a user, usually via a handle, and moved along a desired path to desired target points, to record these points. This approach is much faster and more intuitive as users do not need to be concerned with the different Cartesian coordinate systems. However, this approach is not always feasible, e.g., when the robots are too large. These programming approaches are known as online programming as they require the use of real physical robots, and this is non-productive as tasks cannot be carried out by these robots during the programming process.

[*] Corresponding author.
E-mail addresses: mpeongsk@nus.edu.sg (S.K. Ong), mpewya@nus.edu.sg (A.W.W. Yew), mpenkt@nus.edu.sg (N.K. Thanigaivel).

https://doi.org/10.1016/j.rcim.2019.101820
Received 10 January 2019; Received in revised form 31 May 2019; Accepted 1 June 2019
Available online 06 June 2019

Conversely, offline programming is carried out without the need of the actual robot [2]. Offline programming methods require construction of a robot and a robot workspace in a virtual environment, with teaching and simulating tasks via a virtual model of the robot. Advanced path planning and optimization algorithms with collision avoidance can be employed for offline robot programming, thus reducing the effort required by the human programmers and enhancing the quality of the robot programs. In offline programming, high fidelity CAD models of real robots and their kinematics have to be modelled to represent the physical robots in order to develop realistic simulation. It is time-consuming and computationally intensive to model the physical robot with high fidelity. These models and kinematics may need to be modified repetitively after they have been applied to the real robots [2]. The simulation module needs to be tested on the real robots before actual offline programming can be performed, and these tests may require more effort to refine the models to achieve high fidelity.

To overcome the limitations of conventional robot programming methods, this paper presents an augmented reality-assisted robot programming system (ARRPS). Augmented Reality (AR) has been widely applied to develop applications to provide guidance and/or electronic user manuals to assist users in manufacturing processes, such as maintenance [3] and robot path planning [4–10]. AR, which refers to the overlay of computer-generated virtual objects and user interfaces to a physical environment, can enhance robot programming in two ways. First, AR interfaces can allow users to access computer software and data directly from the physical workspace of a robot, thus enhancing information perception. Second, virtual responsive objects can be overlaid on physical objects to enhance user interaction with the robot programming system. The use of projection-based AR interfaces for robot programming has been reported by Zaeh and Vogl [4] on the use of a laser projector to project an AR user interface, which consists of user-defined end-effector paths and menus, on work surfaces and workpieces for users to edit robot motion parameters. The stylus that is used to interact with the menus on the AR user interface, traces the desired end-effector paths and defines the orientation of the end-effector tracked using an active optical tracking system. A task-level programming AR user interface that uses the stylus to define welding positions, requirements and optimization criteria was added by Reinhart et al. [5]. It is important to note that projection-based AR interfaces are visible only on surfaces. To overcome this problem, Gaschler et al. [6] integrated a computer monitor with video projection to facilitate the visualization of robot tasks in a virtual work cell. This approach, however, causes additional mental load on the users as they have to focus on two different displays, i.e., the display on the monitor and the AR projection in the physical work cell. Fang et al. [7] presented a marker-based AR-assisted system that uses a see-through head-mounted display (HMD) with a handheld pointer to define robot paths and end-effector orientations manually. As this system relies on fiducial markers in the robot work cell for tracking and registration, there are dead-zones in the AR environment where tracking is not feasible; the use of marker-based tracking and registration limits the positions and viewing angles in the robot work cell.

ARRPS is designed to allow users with little robot programming knowledge to program tasks for a serial robot. The system transforms the work cell of a serial industrial robot into an AR environment. The innovative research contributions of ARRPS are as follows:

(1) With an AR mobile user interface and a handheld pointer for interaction, users are free to move around the robot work cell to define 3D points and paths for the real robot to follow. This allows the users to plan in the actual physical robot work cell rather than a fully virtual environment or a mock-up work cell. This also allows the users to observe actual physical obstacles that might be present in the robot work cell and plan and program the paths for the robot to avoid these obstacles. The use of the handheld device provides an intuitive user interface for the users to define 3D points in 3D space during motion planning.
(2) Sensor data and algorithms are used for robot motion planning, collision detection and plan validation. Data from these sensors in the AR environment is obtained in real-time and considered in the collision detection and plan validation algorithms during robot motion planning.
(3) The proposed approach enables fast and intuitive robotic path and task programming, physical context awareness to the users and allows users to focus only on the definition of tasks.

The remainder of this paper will cover the related work in this field, the methodologies, the system design and implementation, and case studies. Current limitations of the system and the scope for future work will be discussed in the conclusion section.

2. Background and related work

The drawbacks and limitations of conventional robot programming methods have inspired numerous research works on the design and development of user interfaces and interaction methods for robot programming.

Projection-based interfaces have been used in industrial robot programming where augmented graphics, such as visualizations of the user-defined points and paths, and other task parameters, are projected directly onto the surfaces of workpieces [4–6,11]. There are two limitations to projection-based interfaces. First, graphical overlays can only be displayed on physical surfaces, such that a physical workpiece must be available. This precludes its use for defining paths away in a 3D space. Second, the types of information that can be provided visually are limited. For example, the pose of the robot end-effector cannot be visualized in a 3D space for the users to adjust. As a result, an additional screen is required to provide additional information visualization and user interaction operations.

Providing information through different display devices is undesirable as it requires the users to fuse data from the different sources mentally. For example, if a robot is displayed as a 3D model in a view isolated from the workspace, it would be difficult for the users to visualize the robot in the physical workspace, and the effects of the motion of the robot on the workspace. Fang et al. [7] developed a path planning system that merged all AR overlays into a single monitor. This allows graphics to be augmented anywhere in the 3D space of a work cell, such that points and paths can be defined in a 3D space. This system utilizes a fixed camera pointing at a robot work cell from an overhead position, to give users an overview of the entire work cell. The transformation between the camera and the robot workspace was obtained by tracking a fiducial marker in the workspace using the camera. However, a fixed camera position limits the perspective of the users. Even if the camera can be shifted, the camera must be able to detect the fiducial marker in its field of view. Several works have used smartphones [12] and tablets [13] for robot tasks programming through AR, such that users can move freely in a work cell. This approach gives users a first person view of the AR environment, making it easier for users to coordinate their movements. However, using a smartphone or tablet for the interface requires at least one hand of the user, making it difficult for the user to physically interact with physical objects in the workspace.

Natural user interaction methods have been investigated for human-robot interaction in an industrial context, such as hand gestures [12], and voice commands [14,15]. However, many research works still favor the use of a handheld pointer that is tracked by a motion capture system [4–6,11], as this method provides a good balance between cost and accuracy of defining positions and orientations in 3D space.

ARRPS provides contextual graphical overlays on a single first-person view of the robot workspace through a HMD. This allows hands-free interaction and freedom of movement in the workspace. A handheld pointer is employed for user interaction at multiple levels of abstraction of control of the robot. At the lowest level, the robot paths can be manually defined by a user using this handheld pointer. Alternatively, the user can define points for the robot to reach via any route between the points determined by the system. At the highest level, a user can select workpieces and workpiece features to define tasks for the robot to perform on certain operations. The AR user interface of ARRPS provides a visualization of the pose and motion of the robot through overlaying a virtual robot on the real robot, as well as contextual interaction options via the handheld pointer.

3. System overview

ARRPS has been formulated and developed to simplify and speed up robot task programming, and enhance user interactions so as to make the robot programming process as intuitive as possible. The development of the system focuses on two aspects. First, AR is used to project the user interface of ARRPS onto the actual workspace, allowing users to interact with the workspace directly, e.g., drawing paths and selecting workpiece features to command a robot to perform tasks. The AR scene consists of graphical augmentation overlaying the physical workspace, and serves to aid users to visualize task plans in progress, task metrics and robot motion. Second, the minutiae of robot programming are hidden from the users as much as possible. Programming processes that are not related directly to the required task, e.g., ensuring smooth joint configuration transitions and maintaining collision-free motion, are handled automatically by the system with the use of depth cameras and motion planning algorithms.

3.1. User-interaction and display

The system utilizes a wireless handheld pointer for the user to define paths and tasks in the workspace of a robot. In this research, the handheld pointer is assembled by attaching a wireless mouse to an aluminum bar that resembles the shape of a robot end-effector (Fig. 1). Infrared reflective markers are attached to the handheld pointer so that it can be tracked by a motion capture system. The handheld pointer serves two purposes. First, it is used as a representation of the end-effector of a robot for the definition of points and paths for the robot to traverse. The tip of the pointer is defined as the tool-center-point (TCP) of the end-effector. Therefore, a target point for the robot to reach can be defined by a user by placing the tip of the pointer at a desired point; likewise, a target path can be defined by moving the pointer along a desired path. The handheld pointer is used as a 3D mouse pointer for interacting with objects in the workspace, e.g., selecting edges of a workpiece and defining the edge as a welding path.

A stereoscopic HMD is employed to display the AR user interface, and provide a view of the augmented workspace, so as to allow users to perceive augmented 3D graphics with depth perception. In this research, an Oculus Rift DK2 HMD is used; two Genius WideCam F100 cameras are used to provide wide-angle side-by-side stereoscopy (Fig. 1). This implementation of the viewing device is inspired by Steptoe [16].

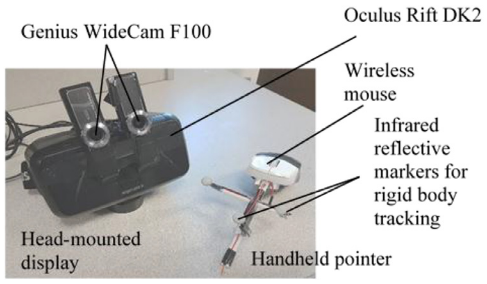

Fig. 1. Interaction and display device used in the proposed system.

3.2. Task planning user interface

A task in ARRPS is composed of individual paths defined by a user and paths that are generated automatically by the system; these paths are visualized in the AR scene to give the user a high-level view of the overall robot motion plan. A user proceeds with task planning by defining paths for a robot using the handheld pointer. The user has to define the critical paths, e.g., a path on the workpiece where the robot will perform certain specific operations such as welding. The via-paths from the start position of the robot to the first user-defined path and the paths in between the individual user-defined paths are generated automatically by the system. This is accomplished using a motion planning library [17] to generate the robot motion between the last point and first point of two successive user-defined paths respectively. The task planning user interface includes contextual menus that display different options according to the types of the entities, i.e., workpiece, point, path, etc., that are selected by the user.

Two types of paths can be defined, namely Cartesian paths and point-to-point paths. For Cartesian paths, a user defines a continuous path for a robot to follow. Thus, the robot motion is constrained to the user-defined sequence of positions and orientations that the end-effector has to follow. For point-to-point paths, a user selects key positions that a robot must reach for a task, while the robot motion between successive points is generated using the motion planning library [17]. In addition to defining paths manually, CAD models of workpieces can be used for defining paths.

To create a Cartesian path, a user presses the button on the handheld pointer and moves the pointer along a desired path. Points at regular intervals along this path are added using the system to form the Cartesian path. A smaller interval length would lead to a greater number of points placed along the path by the system. The pose of each point is based on the pose of the pointer as it is moved through that point along the path. The interval length is a parameter that can be set manually by the user according to the requirements of the application. For example, for robotic welding of workpieces with complex geometries, the interval length can be set to a smaller value so that the user can define more intricate paths. However, this will also result in the capture of the jitter of the user's hand, which can be mitigated by applying smoothing filters to the path. The path is displayed in the AR scene as a line as the user “draws” it with the handheld pointer (Fig. 2a).

Point-to-point paths are created by defining each successive point using the handheld pointer. The pointer is placed at a point in a desired orientation by clicking the button of the pointer. When a user creates a point, a graphical target is drawn at the 3D location in the orientation that reflects the pose of the robot end-effector (Fig. 2b).

Many robotic tasks are carried out with respect to model features, such as points and edges on an object. For defining paths based on CAD models, a user selects a specific feature, e.g., an edge, of the target workpiece in the workspace using the handheld pointer (Fig. 2c). To enable the selection of features using the handheld pointer, physical workpieces are overlaid with their CAD models in the AR scene. In this research, SolidWorks CAD models, which include feature-based parametric models of workpieces, are used to provide the workpiece features in the AR scene.

3.3. Augmented graphical aids

In addition to the task planning interface, the AR scene consists of a virtual robot that is overlaid on the real robot in the workspace. The robot motion parameters for a particular task being planned are displayed in real-time, namely the reachability and manipulability of the robot arm (Fig. 3).

To indicate the reachability of the robot arm with respect to specific points along paths, the virtual robot arm follows the handheld pointer, and the corresponding pose of the links of the virtual robot arm is

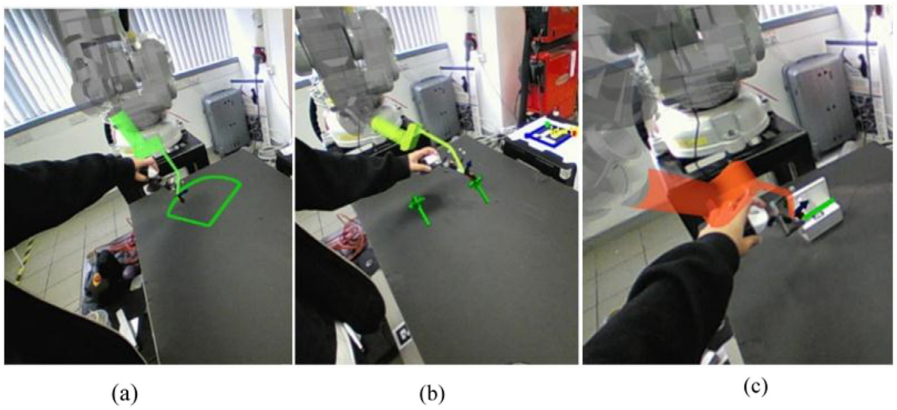

Fig. 2. Robot path programming by defining (a) Cartesian paths, (b) point-to-point paths and (c) selecting workpiece edges.

computed using an inverse kinematics solver. The virtual robot will not follow the handheld pointer through paths that are not reachable.

The manipulability of a robot end-effector, defined as the ease with which the robot is able to move the end-effector in any arbitrary direction from its current position [18], can be visualized using the virtual robot. The manipulability index is augmented to provide an intuitive measure of the manipulability of the robot pose to the users. Four levels of manipulability have been defined, namely, excellent, good, fair and bad, based on the distance of the virtual end-effector to singular robot joint configurations. A "bad" manipulability means that the robot will potentially run into a singular configuration very quickly, while a "good" manipulability means that the robot can be moved in any direction for a large distance before running into a singular configuration. The manipulability is reflected in real-time by the color that is used to display the virtual end-effector in the AR scene. The virtual end-effector displayed in red color indicates very poor manipulability while green means that manipulability is very high. The color varies gradually between the two extremes as the manipulability varies when a user moves the virtual robot. The manipulability indicator is useful for providing a user with an understanding of the difficulty for the robot to reach certain poses. In positions with low manipulability, the robot may not be able to proceed to the next target point; this will result in a high probability that the robot motion that is being planned by a user may reach a dead-end before the task can be completed. In such a scenario, the user will have to cancel the current task plan and repeat the planning of the task using a different path. The manipulability indicator can be used as a quick test of the suitability of workpiece placement as the user can move the robot end-effector to the workpiece to observe the manipulability indicator.

3.4. *Generation of robot motion*

There are a number of issues that must be resolved during the generation of a complete robot motion plan in order to execute all the tasks accurately, quickly and safely. These issues include collisions with other objects in the workspace, and the optimal joint configuration solutions to be selected for a particular robot motion task. To address these issues, vision-based workspace reconstruction methods [19] and motion planning algorithms [17] have been integrated in this research. These will be detailed in the next section. The automatic generation of parameters for specific applications, such as welding or assembly would require specific robot task planning algorithms and expert databases to be plugged into the system. The generation of application-specific robot motion parameters, such as motion speed, will not be studied as it is beyond the scope of this paper.

4. Methodology and implementation

The ARRPS system integrates a HMD and a handheld pointer for robot motion planning; it is supported by motion tracking [20–22] and environment reconstruction technologies [19]. The HMD is connected to a laptop or PC on which the ARRPS software is executed. The functions of the ARRPS software are to render and display the AR scene on the HMD, and track user interactions via the handheld pointer within the workspace. The system consists of a robot motion planning module that provides the functions to compute the inverse kinematics of the robot, generate collision-free motion plans for paths defined by a user and generated by the system, and execute user-defined tasks on a real robot. The system architecture is depicted in Fig. 4.

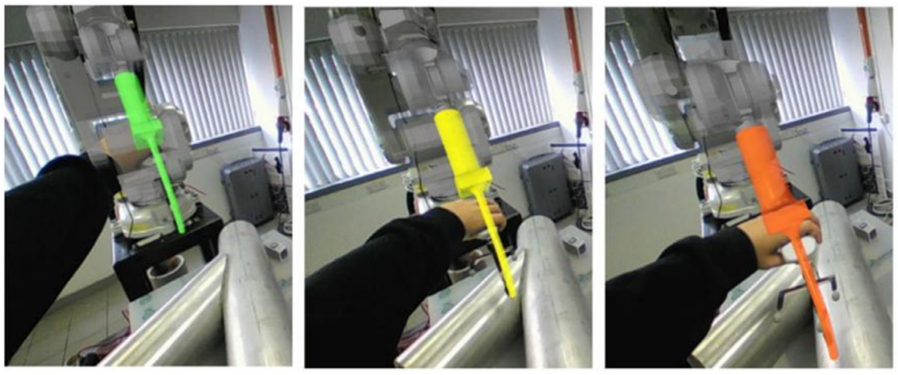

Fig. 3. The reachability and manipulability (good to poor from left to right) of the robot visualized in the AR interface.

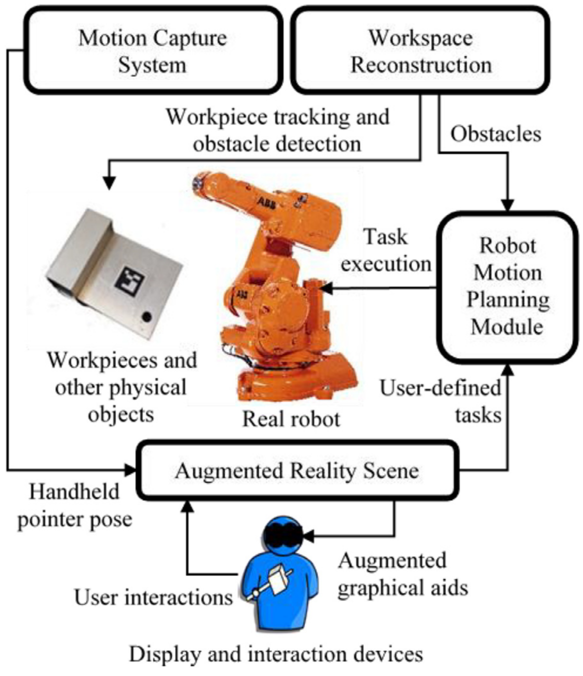

Fig. 4. Architecture of the AR-assisted robot programming system.

4.1. Workspace tracking

The ARRPS system employs three different tracking technologies to track user interactions, the point of view of a user, and physical objects in the workspace. Optitrack, an optical motion capture system, is used to track user inputs via a handheld pointer. The point of view of a user, which must be known in order that the ARRPS software can generate the AR scene, is tracked using a combination of computer vision-based fiducial marker tracking [20] and natural feature tracking [21], which are integrated with the ARRPS software. This combination of tracking techniques allows the pose of the cameras on the HMD with respect to the environment to be calculated. Physical objects in the workspace, which will be used for collision avoidance and workpiece pose tracking, are reconstructed and tracked using a Kinect2 sensor. The Kinect2 sensor is connected to the robot motion planning module. The robot motion planning module uses the point cloud data from the Kinect2 to generate collision-free robot motion plans. The fiducial marker tracking algorithm is implemented in the robot motion planning module and used to calibrate the pose of the Kinect2 with respect to the robot. All three tracking systems are registered to the coordinate system of the robot so that all the tracked components can be represented in the same coordinate system.

A fiducial marker placed in the robot workspace serves as a fixed point in the workspace where its transformation from the robot is known. As the coordinates of the end-effector with respect to the robot coordinate system can be obtained directly from the robot controller, the transformation from the robot to the fiducial marker can be obtained by moving the end-effector of the virtual robot to the four corners of the marker. This provides the coordinates of the corners of the marker with respect to the robot coordinate system, so that the transformation matrix between the marker coordinate system and the robot coordinate system can be solved using the principal component analysis [23].

The transformation of the fiducial marker from the real robot can be obtained accurately. The transformations between the robot and the cameras on the HMD and the Kinect2 is obtained using the fiducial marker tracking algorithm implemented in the ARRPS software and robot motion planning module respectively.

As a user who wears a HMD may move around in the work cell, the fixed fiducial marker may not always be captured by the cameras. Therefore, natural features of the environment are used as tracking features to compute the pose of the cameras. The transformation between the map of the natural features and the robot is obtained through establishing the transformation between the map and the fiducial marker; this is calculated when a tracking result has been obtained from both the natural feature tracking and fiducial marker tracking processes. As the pose of the HMD is tracked using fiducial marker tracking and natural feature tracking methods, the accuracy with which its pose is obtained depends on viewing angles and the distance of the cameras to the markers and features. The HMD tracking error is acceptable as users are generally aware that the paths visualized through the AR interface do not overlay the corresponding physical space exactly.

A more accurate solution based on the Optitrack motion capture system is employed for pose tracking of the handheld pointer. The Optitrack can track rigid objects to sub-millimeter accuracy, such that the ARRPS system can be applied to a wider range of industrial applications. The coordinate system of the Optitrack system is registered to the robot coordinate system through the use of a calibration square that is provided with the Optitrack system. The calibration square is used to set the origin and principal axes of the Optitrack coordinate system. The Optitrack coordinate system is registered to the robot coordinate system using the same method of registering the fiducial marker to the robot. This is achieved by moving the robot end-effector to known points on the calibration square and solving for the transformation matrix. This allows the handheld pointer to be expressed in the robot coordinate system.

The benefit of tracking the handheld pointer and the HMD using separate tracking systems is that the cost of the ARRPS system can be kept low without compromising the accuracy of the user-defined paths. The Optitrack system is an expensive system that can track objects with sub-millimetre accuracy within a tracking volume that is defined by the configuration of its cameras [22]. Thus, by constraining the tracking volume of the Optitrack system to regions where the robot executes tasks, such as worktables and conveyor belts, fewer Optitrack cameras are needed to track the handheld pointer.

4.2. Robot motion planning module

The inputs to the robot motion planning module are the task parameters defined by a user through the AR interface, namely, the points, paths and end-effector orientations for every task. The module generates the corresponding robot motion based on these parameters, which is simulated on the virtual robot and executed on the real robot. This module provides augmentation and visualization of the real-time motion of the virtual robot and other metrics, namely, the reachability and manipulability of the robot, so as to enhance a user decision-making during motion planning for full task execution on a real robot. In the implementation of the ARRPS system, the robot motion planning module is developed based on the Robot Operating System (ROS) so as to make use of the algorithms that are provided by ROS [24].

The motion of a virtual robot is generated based on the path demonstrated by a user through manipulating the handheld pointer. This motion is generated using the inverse kinematics solver implemented in the Kinematics and Dynamics Library (KDL) [25] to compute possible robot joint configurations that allow a point defined by the handheld pointer to be reached by a robot. The pose of the handheld pointer with respect to the robot is streamed continuously to the robot motion planning module. From all the possible robot joint configurations that have been generated using the KDL, the robot joint configuration that is most similar to the current pose of the robot is chosen and the joint angles are computed and applied immediately to the virtual robot. The manipulability index of the virtual robot arm is computed and rendered with the virtual end-effector.

The Open Motion Planning Library (OMPL) [17] is used in the robot motion planning module as a library of motion planning algorithms.

OMPL provides sampling-based motion planning algorithms where motion planning is carried out through sampling the state space of a robot to search for valid robot trajectories. Collision-free motion planning is enabled through integrating OMPL with a motion planning framework that provides geometry representation. MoveIt! [26], which is a motion planning framework that is integrated with OMPL and KDL, is used to represent geometries in the robot workspace as meshes, point clouds or octrees. The workspace is reconstructed as an octree [19] based on point clouds captured by a Kinect2 depth camera. MoveIt! uses an octree to plan collision-free robot motion using OMPL. Thus, ARRPS allows users to provide target points for a robot to reach without having to manually ensure that the paths to reach the target points are collision-free.

5. Application and user studies

A prototype of AARPS has been implemented and applied to two applications, namely, welding and pick-and-place operations. Experiments and user studies have been carried out to evaluate the usability of the system. A total of twenty subjects participated in the user studies. For each application, the user study was conducted with ten participants who were tasked to program the robot to perform certain operations. All the participants had no prior experience in robot programming, and were allowed to be familiarized with the system before commencing the experiment.

For the welding application, each user was required to define the welding path of the robot arm by selecting an edge on the workpiece to be welded (Fig. 5a). The experiment setup was designed such that the workpiece to be welded was in close proximity to other objects, so as to prevent the motion planning algorithm from computing a feasible motion plan to approach and return from the welding path after the welding task is completed. Therefore, users were required to define collision-free approach and departure paths (Fig. 5b).

For the pick-and-place application, users were asked to program the robot to pick up a workpiece on one side of a pipe and place it on the other side of the pipe. The pick-and-place task is defined using the point-to-point path definition operation, whereby only the location and orientation of the gripper at the pick-up location and place-down location need to be defined; the collision-free intermediary paths are generated automatically by OMPL after the workspace has been scanned using the Kinect2 sensor and reconstructed as an octree. Fig. 6 shows the definition of the pick-and-place task using ARRPS.

For every user-defined task, the robot motion was simulated and augmented through the AR interface so that the users could visualize the programmed task. Next, the task was executed using the real robot to verify that the tasks were programmed correctly. The average time to complete the tasks was 63 s and 34 s for the welding and pick-and-place

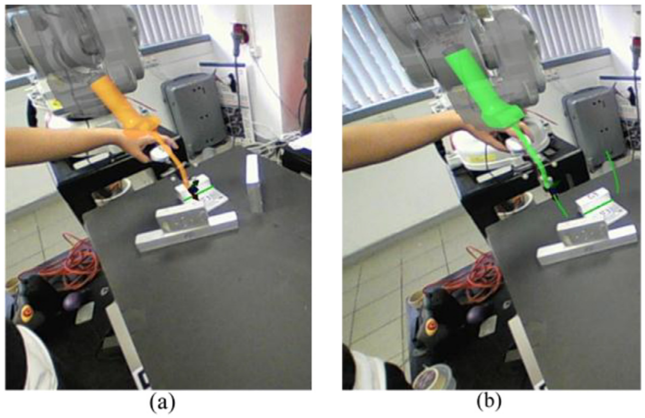

Fig. 5. Programming a robotic welding task by (a) selecting the edge to be welded, and (b) defining the approach and departure paths.

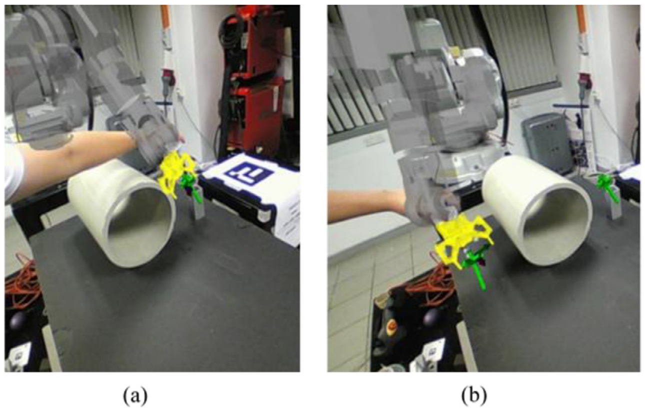

Fig. 6. Programming a robotic pick-and-place task by defining the pick-up (a) and place-down (b) locations with corresponding gripper orientations.

tasks respectively, and the standard deviation of the time taken was 14 s and 8 s respectively. The time to completion for each user was recorded from the time the user began moving the handheld pointer until the completion of a successful simulation of the task as displayed through the AR interface. For comparison, two users who were familiar with robot programming using a teach pendant took an average of 347 s and 117 s respectively to program the two tasks using a teach pendant. The time to completion for programming a task using the teach pendant started from the time the user first started to jog the robot end-effector to the first location and ended upon defining the final location for the robot end-effector to reach; the time to completion does not included testing and validating the robot motion. For the welding task, a significant amount of time was spent on adjusting the orientation of the robot end-effector. For the pick-and-place task, defining the path around the obstacle took up a large portion of the time taken.

The user experience was assessed using a questionnaire based on a 5-point Likert scale (1: strongly disagree; 2: disagree; 3: neutral; 4: agree; 5: strongly agree). The questions are as follows:

a Robotic welding tasks are easy to program using the system.
b Pick-and-place tasks are easy to program using the system.
c The system is easy to use.
d The system is easy to learn.
e I feel confident that the tasks that I program using the system will be executed correctly.
f I feel confident that the tasks that I program using the system will be executed safely.

The results of the questionnaire based on the feedback from the twenty participants are shown in Fig. 7. The results indicate that the user interface of the system is simple for lay-users to use to program industrial robots. The mental load on the users for defining the pick-and-place task is much lighter than the welding task as the actual path of the robot and collision avoidance did not have to be considered by the users. Therefore, the pick-and-place task was found to be easier to carry out than the welding task. However, for both tasks, the users found the AR interface to be user-friendly and intuitive. They felt that the real-time motion visualization and simulation of the task through the AR interface helped boost their confidence as the tasks were being defined correctly and safely.

6. Conclusion and future work

This paper presents the ARRPS system that provides an immersive AR robot work cell for the users and enhances robot path and task programming by providing the users with intuitive methods to define and visualize the robot motion. This system minimizes the need for the

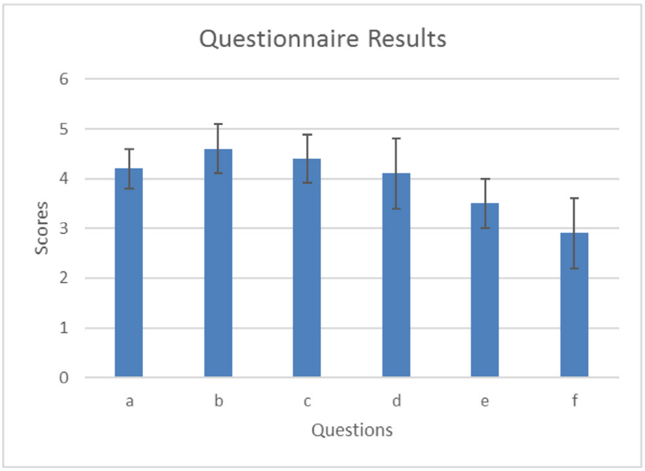

Fig. 7. Results of the user studies questionnaire.

users to handle trivial and tedious motion planning procedures that are not related to the task.

User studies were conducted and the results showed that the system could significantly speed up the programming of robotic tasks and reduce the need for user expertise in robot programming. The results from the questionnaire survey showed that the system was easy to learn and use, and the users felt fairly confident in the tasks that they programmed due to the ability to visualize the robot motion through a virtual robot augmented in the scene of the workspace.

The system can potentially speed up and simplify the process of industrial robot programming in many applications. Further studies are currently being conducted in the integration of ARRPS with sub-millimeter workpiece localization sensors. This integration will allow the system to adjust the user-defined paths, which are prone to noise resulting from hand tremors [27], align these paths accurately with the workpiece joints to be welded, and allow for slight variations in the placement of workpieces without having to reprogram the welding task.

Acknowledgment

This research is supported by the Singapore A*STAR Agency for Science, Technology and Research, Science Engineering Research Council's (1) Industrial Robotic Programme on Interface for Human Robot Interaction, Project no. 1225100001, and (2) Public Sector Research Funding Programme on Embedding Powerful Computer Applications in Ubiquitous Augmented Reality Environments, Project no. 1521200081.

Conflict of interest

None

References

[1] J.O. Oyekan, W. Hutabarat, A. Tiwari, R. Grech, M.H. Aung, M.P. Mariani, L. Lopez-Davalos, T. Ricaud, S. Singh, C. Dupuis, The effectiveness of virtual environments in development collaborative strategies between industrial robots and humans, Robot. Comp. Integ. Manuf. 55 (2019) 41–54.
[2] Z. Pan, J. Polden, N. Larkin, S. Van Duin, J. Norrish, Recent progress on programming methods for industrial robots, Robot. Comp. Integ. Manuf. 28 (2012) 87–94.
[3] M. Gattullo, G.W. Scurati, M. Fiorentino, A.E. Uva, F. Ferrise, M. Bordegoni, Towards augmented reality manuals for industry 4.0: a methodology, Robot. Comp. Integ. Manuf. 56 (2019) 276–286.
[4] M.F. Zaeh, W. Vogl, Interactive laser-projection for programming industrial robots, Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality, Washington DC, IEEE Computer Society, 2006, pp. 125–128.
[5] G. Reinhart, U. Munzert, W. Vogl, A programming system for robot-based remote-laser-welding with conventional optics, CIRP Ann. – Manuf. Tech. 57 (1) (2008) 37–40.
[6] A. Gaschler, M. Springer, M. Rickert, A. Knoll, Intuitive robot tasks with augmented reality and virtual obstacles, Proc. IEEE International Conference on Robotics and Automation, Washington DC, IEEE Computer Society, 2014, pp. 6026–6031.
[7] H.C. Fang, S.K. Ong, A.Y.C. Nee, A novel augmented reality-based interface for robot planning, Int. J. Interact. Design Manuf. 8 (1) (2014) 33–42.
[8] H.C. Fang, S.K. Ong, A.Y.C. Nee, Interactive robot trajectory planning and simulation using augmented reality, Robot. Comp.-Integ. Manuf. 28 (2) (2012) 227–237.
[9] J.W.S. Chong, S.K. Ong, A.Y.C. Nee, K. Youcef-Youmi, Robot programming using augmented reality: an interactive method for planning collision-free paths, Robot. Comp.-Integ. Manuf. 25 (3) (2009) 689–701.
[10] S.K. Ong, J.W.S. Chong, A.Y.C. Nee, A novel AR-based robot programming and path planning methodology, Robot. Comp.-Integ. Manuf. 26 (3) (2010) 240–249.
[11] D. Antonelli, S. Astanin, Qualification of a collaborative human-robot welding cell, Procedia CIRP 41 (2015) 352–357.
[12] J. Lambrecht, J. Kruger, Spatial programming for industrial robots based on gestures and augmented reality, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Washington DC, IEEE Computer Society, 2012, pp. 466–472.
[13] J.A. Frank, M. Moorhead, V. Kapila, Realizing mixed-reality environments with tablets for intuitive human-robot collaboration for object manipulation tasks, Proc. 25th IEEE International Symposium on Robot and Human Interactive Communication, NY, USA, 2016, pp. 302–307.
[14] J.N. Pires, G. Veiga, R. Araujo, Programming-by-demonstration in the coworker scenario for SMEs, Indust. Robot Int. J. 36 (1) (2009) 73–83.
[15] B. Akan, A. Ameri, B. Çürüklü, L. Asplund, Intuitive industrial robot programming through incremental multimodal language and augmented reality, Proc. 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3934–3939.
[16] W. Steptoe. (2013). AR-Rift. [Online]. Available:http://willsteptoe.com/post/66968953089/ar-rift-part-1.
[17] I.A. Şucan, M. Moll, L.E. Kavraki, The open motion planning library, IEEE Robot. Automat. Mag. 19 (4) (2012) 72–82.
[18] T. Yoshikawa, Manipulability of robotic mechanisms, Int. J. Robot. Res. 4 (2) (1985) 3–9.
[19] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: an efficient probabilistic 3D mapping framework based on octrees, Autonom. Robot. J. 34 (3) (2013) 189–206.
[20] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, M.J. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recog. 47 (6) (2014) 2280–2292.
[21] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, Proc. 6th IEEE/ACM International Symposium on Mixed and Augmented Reality, Washington DC, IEEE Computer Society, 2007, pp. 1–10.
[22] OptiTrack Flex 3. (February 2017). [Online] Available:http://optitrack.com/products/flex-3/indepth.html.
[23] P.J. Best, N.D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (2) (1992) 239–256.
[24] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, Proc. 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009, pp. 5–10.
[25] R. Smits. (February 2017). KDL: kKinematics and dynamics library[Online]. Available:http://www.orocos.org/kdl.
[26] I.A. Şucan, and S. Chitta. (2017). MoveIt!. [Online]. Available:http://moveit.ros.
[27] D. Antonelli, S. Astanin, M. Galetto, L. Mastrogiacomo, Training by demonstration for welding robots by optical trajectory tracking, Procedia CIRP 12 (2013) 145–150.