PoinTap system: a human-robot interface to enable remotely controlled tasks

PoinTap system: a human-robot interface to enable remotely controlled tasks

Fiorella Sibona, Pangcheng David Cen Cheng, Marina Indri, Danilo Di Prima
Dipartimento di Elettronica e Telecomunicazioni (DET)
Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy
{fiorella.sibona, pangcheng.cencheng, marina.indri}@polito.it, danil.diprima@gmail.com

***Abstract*—In the last decades, industrial manipulators have been used to speed up the production process and also to perform tasks that may put humans at risk. Typical interfaces employed to teleoperate the robot are not so intuitive to use. In fact, it takes longer to learn and properly control a robot whose interface is not easy to use, and it may also increase the operator's stress and mental workload. In this paper, a touchscreen interface for supervised assembly tasks is proposed, using an LCD screen and a hand-tracking sensor. The aim is to provide an intuitive remote controlled system that enables a flexible execution of assembly tasks: high level decisions are entrusted to the human operator while the robot executes pick-and-place operations. A demonstrative industrial case study showcases the system potentiality: it was first tested in simulation, and then experimentally validated using a real robot, in a laboratory environment.**

***Index Terms*—Industrial manipulators, human-robot interface, assembly tasks, remote control**

I. Introduction and state of the art

Industrial manipulators have been widely used during the last decades. They are typically employed for carrying out automated repetitive tasks that may be dangerous or risky to humans, such as those carried out at high speed or with heavy objects involved. Depending on the application, there are many scenarios in which a robotic arm can be used. It can be fixed to a workstation to perform repetitive tasks, to work along with a human operator or to be remotely controlled if the environment is dangerous for human beings. In the latter case, a Human-Robot interface is needed to properly monitor and control the robot actions. A typical device to manually control industrial manipulators is the Teach Pendant, whose interface is similar to a joystick. However, this kind of interface is not very intuitive, and it may require a lot of practice in some applications [1].

In [2], a cyber-physical system is implemented to remotely control a robot in hazardous manufacturing environments. In particular, the human operator can control the physical robot located in a remote workstation either using a virtual model of the system or guiding a collaborative manipulator in a human-robot workstation. In the second case, a wearable display device is used to give a visual feedback of the teleoperated robot to the human operator.

A robot can also be teleoperated using approaches exploiting typical human to human communication interfaces, such as speech or gestures. Industrial environments are usually very noisy, so it is difficult to operate the robot using vocal commands, and gestures are preferred over speech [3]. Moreover, gestures can be easily executed with only one hand, by articulating a sequence of fingers and hand movements and therefore, they are also easy to understand. A common gesture used in human-human interaction is to point a part, a location or indicating a position. This is as simple as it is useful to codify real-world assembly tasks [4].

There exist other interfaces for teleoperation that may be adopted to ensure a quick response, which improves usability. In fact, in [5], it was seen that people perceive more accurately and prefer a system they are more comfortable with, such as a touchscreen. Nowadays, most of the human-robot interfaces include a touch screen, becoming easy and intuitive to use [6]. It is worth noting that in an industrial environment, human operators may wear gloves for safety reasons, so the Teach Pendants with a touch screen interface may not be convenient, since the screen is relatively small. A possible solution to such problem could be that of enlarging the touch screen device. Nevertheless, this approach could be not convenient, as it cannot be adopted if the operator wears gloves and large touch screen devices are usually expensive. In [7], an infrared-matrix sensory system is used to emulate a touch screen interface that recognizes the user's actions and therefore, to control the manipulator remotely. Four infrared matrices are attached to the corners of a display, in such a way that the infrared transmitters and receivers can detect correctly the (x, y) coordinates of the fingers touching the screen.

The quality of the interaction can be improved when there exists a confirmation feedback within the human-robot communication, as shown in [8]. It was demonstrated that the selection accuracy while performing a task is higher than in absence of any feedback. In a similar way, gestures can be used in conjunction with visual feedback markers. However, this approach often requires expensive devices due to the fact that it generally works with Augmented Reality (AR) techniques.

In [9], an AR system is integrated to the communication loop between humans and robots with the aim of enhancing the human-robot interaction. In this framework, human operators wear the Microsoft HoloLens glasses to receive real-time information related to the tasks, and use the air tap gesture to command the robots and to provide a feedback to the main control system. Similarly in [10], AR is used to improve the

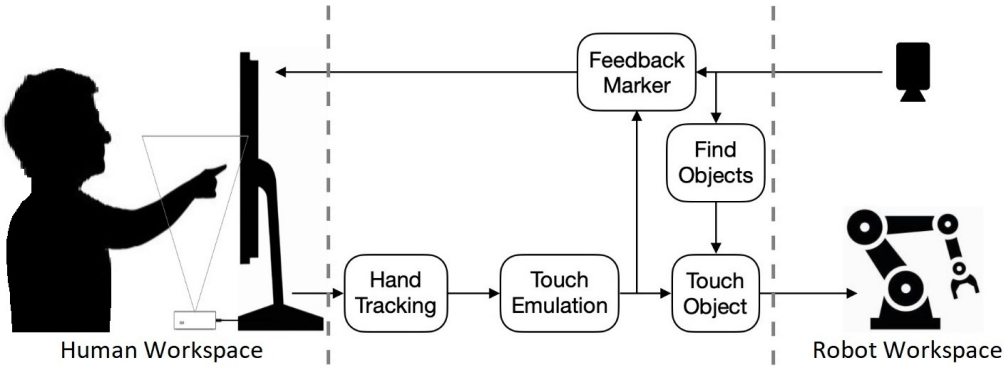


Fig. 1: Block diagram of the proposed system.

robot teleoperation experience, providing a virtual foresight of the real robot trajectories and final location.

The scenario that will be considered in this work is a production environment that does not allow the presence of human operators due to safety reasons but, however, requires the robotic system to be controlled onsite. In such cases, the sensors used for monitoring the activity of the robot and receiving the commands from the human operator are fundamental to achieve an efficient communication between humans and robots for teleoperation tasks [11].

Recently, a considerable literature has grown up around the theme of hybrid make-to-stock/make-to-order manufacturing strategies [12]; make-to-order manufacturing involves high level decisions aiming at the customization of final products, given the market demand that sees an increasing trend towards customized goods production, and a consequent evolution of the production line [13]. When some functional features are available on the robot's system, e.g., low level planning, sensory system to monitor the workspace, status update and the possibility to receive tasks inputs from the operator, it is possible to command the robot with high level tasks [14]. An interface that allows to control the robot with high level tasks reduces the human operator workload, as demonstrated in [15], [16], since the worker does not intervene in the motion of each joint, but instead the robot is able to compute the shortest trajectories for carrying out the operations.

This paper aims at showing an alternative way to remotely control a robotic arm to perform assembly tasks. The proposed system emulates a touch screen behaviour exploiting a hand-tracking sensor along with a liquid-crystal display (LCD). The latter is also employed to show a visual feedback to the operator. A monocular camera streams the workspace where objects to be identified and grasped lie. The remaining of this paper is organized as follows: Section II illustrates the main conceptual elements of the proposed PoinTap interface. Then, with the aim of describing the functionalities of each block, the PoinTap system implementation and testing applied to a specific case study are unfolded in Sections III-B and IV, respectively. Finally, Section V draws some conclusions and identifies some open issues.

II. PoinTap Interface – Concept

The proposed interface aims at providing an innovative combination of resources and devices to enable human-controlled actions on a manipulator. In particular, the PoinTap live feedback allows to improve the interaction experience of the human operator, while exploiting a simple LCD screen. The latter is upgraded to an emulated touchscreen, externally enabled using a hand-tracking device, inspired by the work presented in [17]. The present work enriches the emulated touchscreen with the possibility of pointing to the screen while receiving a live visual feedback. The whole system is imagined as a human-robot interface to perform conjoined actions within an industrial context.

Figure 1 shows the PoinTap interface system architecture in the form of blocks: arrows highlight the data flow, clarifying the input information combinations for each output. As can be seen, the main system blocks serve as interface between the human workspace and the robot workspace.

Given the analysed state of the art, the PoinTap interface encapsulates previous works' features to implement the envisioned interface on an LCD screen. The screen shows the robot workspace scene, provided by a top-view camera, based on which the human operator interacts through a "point at part" gesture, fed to the *Hand Tracking* and *Touch Emulation* (TE) blocks. The latters interpret this information and output it as a feedback mark on the screen. The system verifies that the pointed/tapped area on the streamed image is actually corresponding to a recognized object and transforms its coordinates into coordinates interpretable by the robot. Such a region is then identified as an area of interest for a desired task by the robot. The term PoinTap, introduced to denote the proposed solution, is just related to such a capability of translating the operator pointing/tapping gesture. While Figure 1 helps distinguishing the flow of information among input and output elements of the developed system, a top-down description comes in handy to deepen the role of each single block. We can see the PoinTap system as composed of several layers, schematically arranged in Figure 2. In the extremities (top and bottom) of the scheme, the input and output devices can be found, while the software is in the middle. Note that

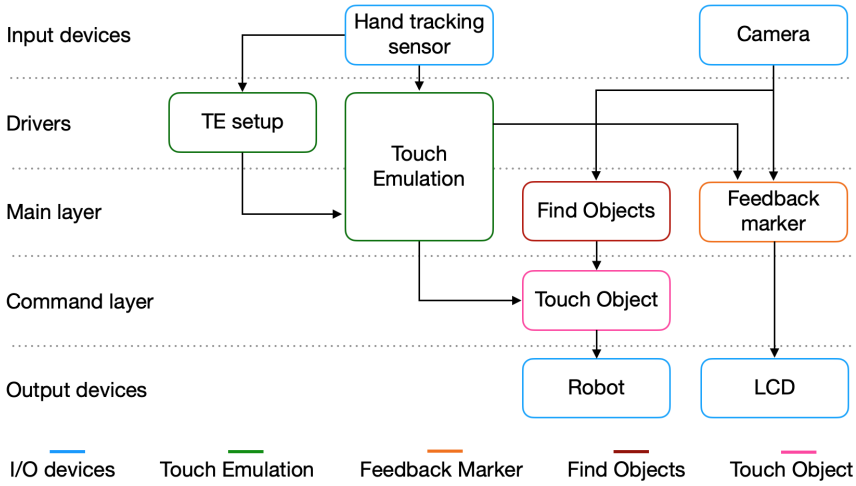


Fig. 2: Schematic representation of the proposed system and its division in layers.

information flows from the top (input devices) to the bottom (output devices).

The role and the actions of the various layers are described hereafter, as the information flows into each of them:

Input devices: a hand-tracking sensor is in charge of recognizing human hands, while a camera positioned on top of the manipulator gathers images of the robot workspace.

Drivers: the software tools in this layer are used as links between the input devices and the main layer. They perform a set-up phase to handle raw information to make it available to the subsequent layer.

Main layer: this layer represents the core of the system. It is responsible for: (i) the visualisation and manipulation of images coming from the camera, (ii) the implementation of the touch emulation system, and (iii) the detection and recognition of objects in the robot workspace.

Command layer: this layer fuses together information that comes from the user and the robot workspace. It takes the information provided by the *Touch Emulation* block and the one coming from the *Find Objects* block, and verifies if the combination of the two leads to an intersection, i.e., a touched area actually corresponds to an identified object. If the latter is true, then the current layer sends the command to the final one.

Output devices: the robot and the LCD screen, which shows images coming from the camera and the live feedback mark, are considered as output devices.

It is worth noting that several elements of the system can be customized, to adapt the generic system to both the available resources and the specific application. Table I highlights the PoinTap elements that can be adjusted to the specific industrial application.

TABLE I: PoinTap Blocks and relative features.

Block	**Features**
Hand Tracking, Touch Emulation (TE), TE Setup	The hand-tracking sensor can be freely chosen: the implementation of the Touch Emulation block and its setup are not strictly interlinked to a specific sensor. Indeed the information needed as input is the position of the tip of the index, independently of the sensor that provides it.
Camera	The camera sensor choice is independent of the other blocks. A custom choice can be made, considering that the camera specifications affect the quality of the streamed image of the robot working space.
Find Objects	This block's implementation depends on the selected object recognition tool.
Touch Objects	This block should be used as it is. Some adaptations may be needed, depending on the software choices for the previous and next blocks.
Feedback Marker	This block strongly depends on the framework used to implement the PoinTap Interface.
Robot	The manipulator can be chosen according to the requirements specific to the application or depending on the available configuration.
LCD	The LCD screen specifications (length, width and height) are free to be chosen, as they are saved as parameters. The only requirement is that the screen lies inside the field of view of the chosen hand-tracking sensor.

With the goal of illustrating the PoinTap interface implementation, complete of the customisable blocks, a case study is considered in the next Section.

III. PoinTap Interface – Implementation

In order to provide a more detailed description of the PoinTap system implementation, a possible use case in the industrial context has been selected. The choice fell on one of the most common applications for a manipulator, i.e., an assembly task achieved by performing a sequence of pick and place operations. The human operator can now see the robot and a set of pieces on the screen. By using a "point at part" gesture, the user points at the piece of interest and, guided by the feedback mark, taps the corresponding area on the screen. This is interpreted by PoinTap, which passes the information to the robot as corresponding to the first object used in the assembly task. This operation is performed one or more times to choose the other pieces for the assembly task. The main results obtained during the validation (both in simulation and with a real robot) of a specific example of the envisioned industrial use case are illustrated in Section IV.

A. Software and Hardware

For what concerns the implementation of the PoinTap system, the Robot Operating System (ROS) framework [18] was chosen. Moreover, as mentioned before, the touch emulation block was inspired to the already available Layered Touch Panel [17], leading to the choice of the Leap Motion (LM) sensor [19] as a hand-tracking sensor. The image stream has been provided by a C210 Webcam by Logitech [20] which has a resolution of 640 x 480 pixel, and is small and inexpensive. The objects recognition task was entrusted to the `find_object_2d` [21], a ROS package which integrates the Find-Object application [22] provided by OpenCV [23] with the image stream coming from the webcam to implement SIFT, SURF, FAST, BRIEF and other feature detectors and descriptors to detect and recognise images from a pre-recorded database. The ROS visualization tool, *rviz*, was exploited for the live feedback mark visualization, while the well-established integration with the Gazebo simulator [24] allowed for a smooth integration of the interface with a simulated version of the chosen robot manipulator.

The implemented interface, even if imagined for the mentioned industrial use case, was tested on a research manipulator, given the possibility to validate it in a laboratory environment. To this end, a Niryo One manipulator [25] was employed for the showcased example. Niryo One is a 6-axis desk robotic arm, conceived for educational and research purposes. It is Open Source and 3D printed, favouring its use as a low-cost option for technology prototyping and validation. Among the available end effectors, the "Gripper 1" was used. It is worth mentioning that, to improve performances, it was decided to distribute the system among several machines: the simulation software was run on a PC, the camera and vision nodes were executed on a Raspberry Pi (RPI) board [26], and a further RPI board ran the nodes in charge of interfacing and controlling the robotic arm, adapted to our case. A summarizing schema of the blocks implementation of the PoinTap system is reported in Figure 3.

Note that blocks in the layers going from the *Input layer* to the *Command layer* (refer back to Figure 2) can be considered in common between the simulated version of the developed case study and its execution on the real robot. With the aim of providing a linear description of the implementation, the main PoinTap blocks (refer back to Figure 1) are illustrated in the next section, following the previously described layered organization.

B. Implementation

As shown in Figure 3, the human workspace is composed of the LCD screen (serving both as an input and an output device) and the LM sensor, both placed on a flat surface. The latter was considered to be perpendicular to the plane where the LCD screen lied, as it is quite common to have the described configuration on a working desk. Note that, beyond its use as output device to display images coming from the camera and the feedback marker, the LCD screen was used also as a reference to define the touch and virtual panels. Furthermore, it is worth recalling that the system implementation does not depend on the chosen screen specifications, with the only requirement that the latter lies inside the LM sensor field of view.

For what concerns the robot workspace, the robot was positioned at the center of the world frame while the camera was placed above it, in such a way to visualise the whole workspace. Moreover, the image plane axes were set to match the plane where the robotic arm and pieces are placed, so as to have a direct correspondence of coordinates.

1) Touch Emulation (TE): the development of this block was led by the identification of the following problems in gesture recognition:

- Recognizing a pointing finger gesture may require the operator to achieve a high level of precision, resulting in an unnatural gesture execution. Achieving a precise gesture can be unpractical for the user, especially when the task consists of a repetition of pick and place operations.
- Timing associated to the pointing gesture is relevant. Indeed, a tradeoff must be identified to allow recognition in an acceptable time while letting the user perform the supervised assembly task.
- It is difficult to identify a "base point". The connection of the latter with the "tip point" (corresponding to the tip of the index) defines the pointing line, which intersects the pointed plane. In this case, as the user's head cannot be detected by the hand-tracking sensor, it could not have been used as the base point. On the other hand, if the base of the finger were taken as base point, the resulting pointing line would mismatch what potentially expected by the operator, making the interaction unnatural.

The TE system was designed to be independent of the above problems and to ease the human operator interaction by enriching the system output with a feedback marker. The TE system proposed in this work provides a 1:1 scale among the scene image and the LCD screen, which corresponds to the touching plane. Exploiting the concepts related to the Layered

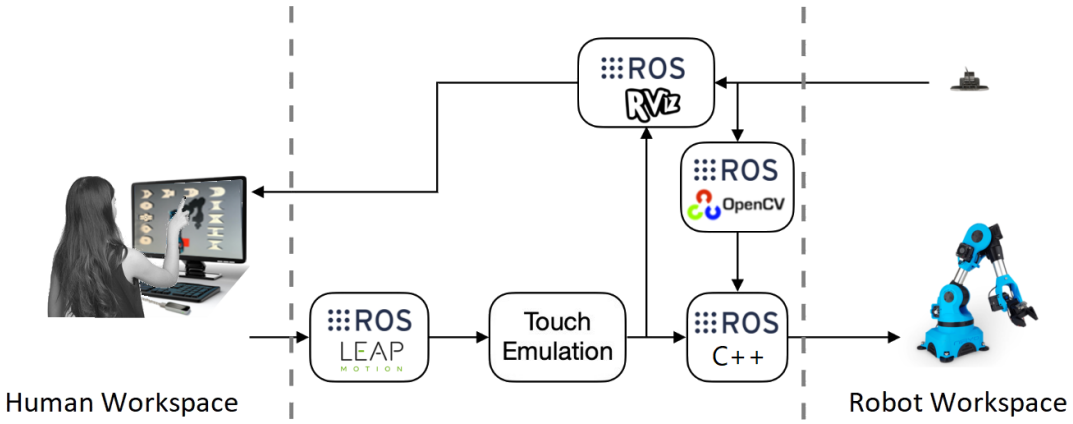


Fig. 3: Block diagram of the PoinTap system implementation.

Touch Panel [17], the real time feedback was implemented by adding a virtual panel to the LCD (Figure 4 shows the virtual screen representation in *rviz*).

(i) *TE setup block* – A calibration phase is needed in order to use the TE block. This procedure, to be performed just once as soon as the system is set up, consists of the screen definition using the Leap Motion. The user touches with his/her index each screen edge, starting from the top left corner of the screen, and saves its coordinates by pressing the spacebar or the enter key. The edges positions in space are processed to obtain the screen parameters (dimensions, position and orientation with respect to the world frame) that are saved on a `.yaml` file, which loads them on the ROS Parameter server at each system start. Note that the screen orientation, saved as the normal vector entering the screen, is provided by the assumption that the screen plane is perpendicular to the plane on which the hand-tracking sensor is placed.

(ii) *TE block* – After the calibration phase is completed the TE block is ready to work. Inspired by the Layered Touch Panel, in the TE block two virtual planes parallel to the LCD were defined, at a distance equal to the *touching distance* and the *hovering distance*, respectively. The first virtual panel is used to capture a touching event, the second for the hovering event. A schematic representation of the considered virtual panels is shown in Figure 5.

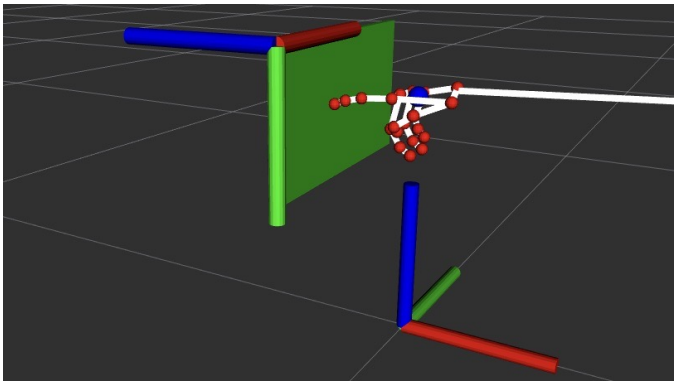


Fig. 4: Virtual representation of the screen in *rviz*.

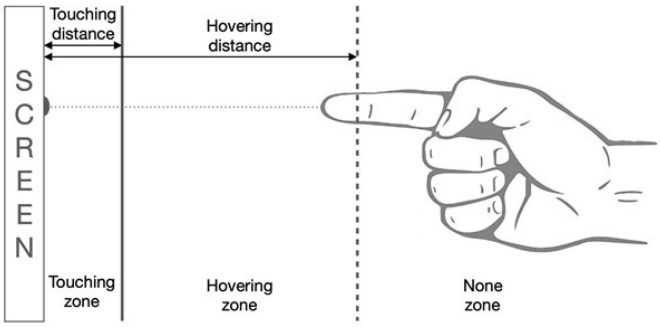


Fig. 5: Graphical representation of the two virtual panels.

The hovering and touching distances are received as input, along with the the LM data and other parameters, e.g., the image dimension expressed in pixels and the pointer radius of the marker visualised on the screen.
At the *Drivers* layer, the screen parameters are used to create a new coordinate system to be associated to the screen and to render its visualization. It was decided to publish the information about the position and the orientation of the screen at a very low rate, as it was assumed that the screen and the hand-tracking sensor do not move with respect to the world frame.
Then, within the *Main layer*, the position of the tip of the index is transformed from the hand-tracking sensor's coordinate system to the screen coordinate system. Given this information, the following conditions are checked:

- If the distance from the screen is less than the hovering distance, a hovering event is triggered and a message containing the position of the tip of the index is published on the dedicated ROS topic, otherwise nothing happens.
- If the tip of the index is in the hovering zone, another condition needs to be checked: if the distance from the

screen is smaller than the touching distance, then the touching event is generated and a message, containing the index tip information, is published.

It is, of course, easier to verify these two conditions while working in the screen frame, since only the values on the z axis are compared.

2) Feedback Marker: As mentioned before, feedback to the pointing gesture is usually implemented using expensive devices for AR. In this work, live feedback was integrated using low cost devices. Indeed, the LCD is used to (i) show the real workspace and, in conjunction with the LM hand-tracking sensor, emulate a touchscreen, and (ii) display a feedback marker in correspondence of the finger tip. The Feedback Marker block receives as input the image stream coming from the camera and the hovering position sent by the TE block. When an image is available, the block simply displays it or, if it receives a message containing the hovering position from the Touch Emulation block, a virtual marker is added to the image.

The marker is represented by a black circle having radius equal to a previously defined variable and it represents the point that the finger is going to touch. The position of the tip of the index is scaled to the screen range and converted into pixels in order to have a direct correspondence. It is worth noting that in this case, the position of the marker is simply the normal projection of the fingertip on the screen, unlike previous studies in which it was given by the intersection of the plane and the pointing direction. This decision was made to give the user the freedom to touch the screen not only using the fingertip but also using the finger pad, which would lead to project the virtual marker far from the fingertip if the pointing direction was used.

3) Find Objects: The Find Object block receives the image stream provided by the camera and a set of photos to be recognised in the workspace. In particular, at setup time, using the Qt based GUI provided by the `find_object_2d` package, the user can load some objects that need to be recognized in the form of locally stored images or taking photos of the current workspace, as in this case. Note that images of the *simulated* objects where used for recognition, both in the simulated case and the experimental testing (further details about the obtained results are available in Section IV). As an object is recognised, a coloured polygon is drawn around it and an unique numerical ID is assigned to it. Moreover, the block generates in output a message containing the positions of the recognised objects with respect to the whole image, as soon as the application recognises a variation in the number of recognised objects.

4) Touch Object: The Touch Object (TO) block receives as input the data generated by the Find Objects and the Touch Emulation blocks. In particular, when the message containing the positions of the identified objects is published by the Find Object block, the TO block stores the information in an internal variable. Then, as a new message containing the touch position is made available by the TE block, the TO block verifies if the position of the tip of the index corresponds to recognised object and, if the latter is true, it generates as output a message containing the ID of the object that has been touched. This information is then read by the robot and used for the task execution. For simplicity, the condition to be checked is considered to be satisfied if the index's tip position is inside the rectangle inscribed in the polygon.

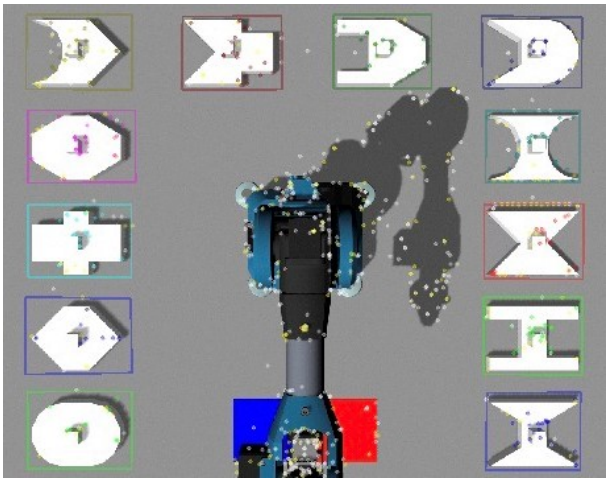
IV. PoinTap Interface – A Case Study

In this section we report the results obtained during the simulation and experimental validation. Consider that, given the use of the ROS framework, the described PoinTap implementation was valid for both the simulated case and the experimental one: the ROS topics and nodes mechanisms allowed for a smooth transition from the simulated case study to the real one by simply substituting simulation nodes with the ones needed to interface the real robot. As previously anticipated, the envisioned use case sees the human operator exploiting the PoinTap system in order to give high-level commands to the manipulator in an assembly task. The idea is that pieces are chosen following some criteria that the manipulator is not able to evaluate (e.g., choices are influenced by customized product requirements).

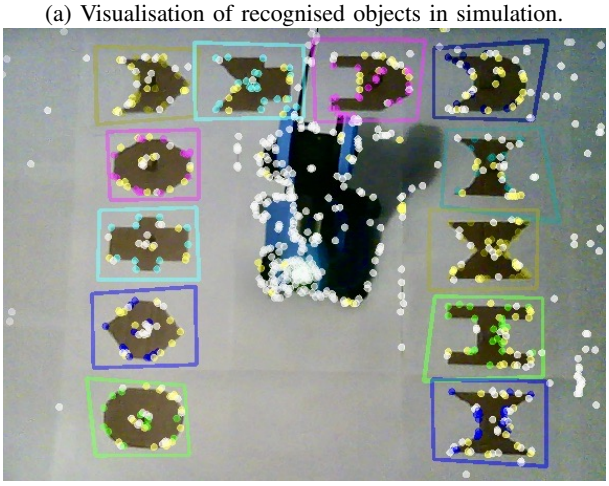
The human workspace was set up with an LCD screen positioned at a fixed pose with respect to the Leap Motion device. For what concerns the robot workspace, twelve pieces were placed around the robot along a horseshoe shape. Specifically, the pieces used within the simulated environment (in Gazebo) were reproduced as accurately as possible using cardboard, to perform the experimental validation. As shown in Figure 6, displayed objects were enclosed in coloured polygons, indicating they were correctly recognized. In particular, Figure 6a shows the twelve pieces in simulation, while Figure 6b reports the corresponding configuration in the real-world scenario. The pieces were numbered from 1 to 12 starting from the bottom-left corner then moving clockwise, as the ID related to each piece. As the human operator "point-tapped" the first object on the touch-emulated screen, the feedback marker appeared as shown in Figure 7.

Keeping in mind the structure in Figure 3, it is worth pointing out that the simulated and real tests share the human workspace configuration, the devices within it, and the core PoinTap system including the robotic arm control and pick-and-place planning nodes. Indeed, simulation only involved the robot workspace and the devices within it, i.e., the robotic arm and the camera.

First, the PoinTap system was tested within the simulated environment. To perform the assembly task, two pieces were selected by touching the screen and then, the simulated robot picked and placed them in a dedicated space for the assembly to happen. As soon as the first object (piece number 8, top-right corner) was point-tapped by the operator and identified by the system, the simulated Niryo One received the command and performed the first pick-and-place operation, then waiting for the second object to be touched (piece number 5, top-left corner). As the second piece was recognized and selected by



(a) Visualisation of recognised objects in simulation.



(b) Visualisation of recognised objects in the real-world.

Fig. 6: Visualisation of the object recognition using the `find_object_2d` GUI.

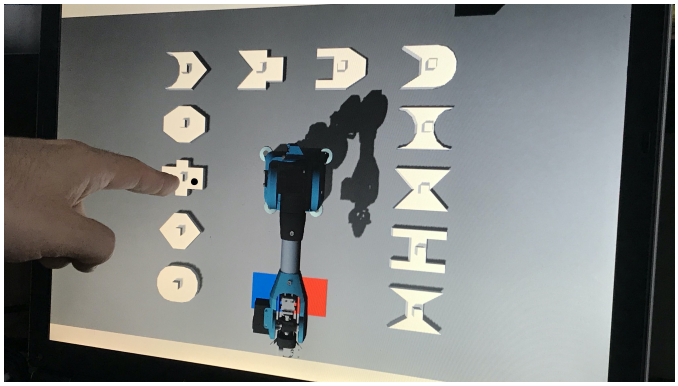


Fig. 7: Representation of immediate feedback (black circle) used in this system, for the simulated case study.

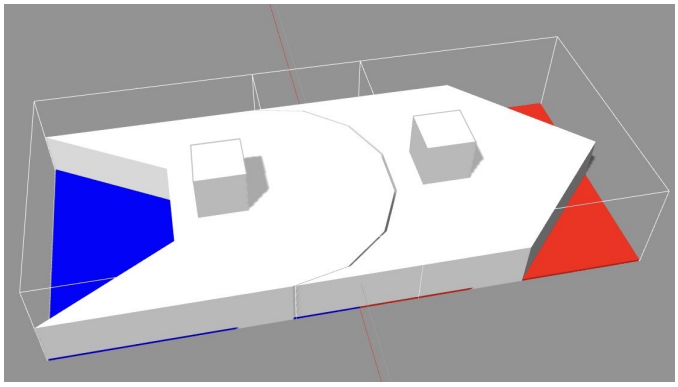


Fig. 8: Representation of the completed assembly task in the Gazebo world.

touching it, the robot grasped and placed it next to the previous piece, as shown in Figure 8.

Then, with the aim of validating the algorithm, two different pieces were point-tapped (hence, selected) by the operator to let the real robot execute the operation. Considering the same pieces' configuration as in the simulation, the first object to be grasped was piece number 5, followed by piece number 10, as shown in Figure 9. The experimental validation results can be seen at [27].

V. Conclusions and Future Works

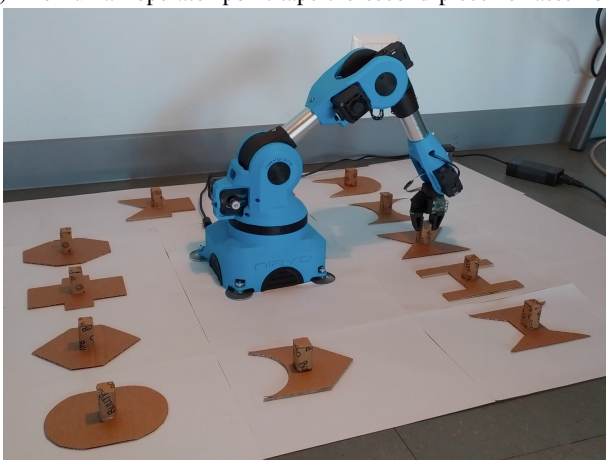
In this paper an interface to remotely control a robotic arm was presented, in particular for supervised pick-and-place operations within an industrial assembly task.

The PoinTap system can enhance flexibility when a re-organization of the assembly task is required. Indeed, the sequence of objects to be assembled can be intuitively selected by point-tapping the LCD screen, without the need of trained operators. Furthermore, as pointed out, in an industrial environment, human operators usually wear gloves for safety reasons, so they are not able to interact with touchscreen devices: our approach may overcome this limitation. Moreover, since the proposed system has different functional blocks, the latters can work independently from each other, and consequently be substituted with other blocks whose functions are similar. Besides, given the fact that pieces are recognized through software, their shape is not relevant, as it is sufficient to train the object recognition system.

It is worth mentioning that, since the touch panel distance from the screen is tunable, PoinTap can be useful to avoid touching the screen so as to improve hygienic conditions. This is a topic of interest when considering health emergency situations such as the one we have and are witnessing.

However, the system presents some weak points to be subsequently addressed. First of all, hand-tracking sensors have a limited field of view so, depending on the screen size, two or more sensors may be needed in order to cover the necessary surface for a correct TE implementation; the need for an additional sensor would lead to further tuning in order to perform data fusion. Moreover, since a monocular camera was used to monitor the scene, it is not possible for the robot to be aware of the actual position of each piece, and hence, if some pieces are initially placed in positions different from the expected ones, the robot may not be able to grasp them.

(a) The human operator point-taps the second piece for assembly.



(b) After the touched object is identified, the robot picks and places it, to complete the assembly.

Fig. 9: Pick-and-place procedure in the real-world scenario.

The PoinTap system could be improved equipping the robot workspace with additional vision sensors, with the aim of retrieving the objects' positions. This would allow to correctly plan the needed trajectory to reach the object even if its position is slightly different from the expected one. Such upgraded vision system could also improve the robot accuracy and precision when pieces are released. Additionally, depending on the screen size, different positions for the hand-tracking sensor could be evaluated in order to cover the largest possible area and boost hand-tracking capabilities. Finally, to customize the assembly task, the range of input commands could be extended, e.g., enabling the rotation of pieces or adding the possibility to cancel or abort an operation.

REFERENCES

[1] K. Krot and V. Kutia, "Intuitive methods of industrial robot programming in advanced manufacturing systems," in *International Conference on Intelligent Systems in Production Engineering and Maintenance*. Springer, 2018, pp. 205–214.

[2] H. Liu and L. Wang, "Remote human–robot collaboration: A cyber–physical system application for hazard manufacturing environment," *Journal of manufacturing systems*, vol. 54, pp. 24–34, 2020.

[3] G. H. Lim, E. Pedrosa, F. Amaral, N. Lau, A. Pereira, P. Dias, J. L. Azevedo, B. Cunha, and L. P. Reis, "Rich and robust human-robot interaction on gesture recognition for assembly tasks," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2017, pp. 159–164.

[4] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, and J. Alcazar, "Gestures for industry intuitive human-robot communication from human observation," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 349–356.

[5] A. Dünser, M. Lochner, U. Engelke, and D. R. Fernandez, "Visual and manual control for human-robot teleoperation," *IEEE computer graphics and applications*, vol. 35, no. 3, pp. 22–32, 2015.

[6] J. Berg and S. Lu, "Review of interfaces for industrial human-robot interaction," *Current Robotics Reports*, vol. 1, no. 2, pp. 27–34, 2020.

[7] S. Bier, R. Li, and W. Wang, "A full-dimensional robot teleoperation platform," in *2020 11th International Conference on Mechanical and Aerospace Engineering (ICMAE)*. IEEE, 2020, pp. 186–191.

[8] C. P. Quintero, R. Tatsambon, M. Gridseth, and M. Jägersand, "Visual pointing gestures for bi-directional human robot interaction in a pick-and-place task," in *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2015, pp. 349–354.

[9] N. Kousi, C. Stoubos, C. Gkournelos, G. Michalos, and S. Makris, "Enabling human robot interaction in flexible robotic assembly lines: An augmented reality based software suite," *Procedia CIRP*, vol. 81, pp. 1429–1434, 2019.

[10] M. E. Walker, H. Hedayati, and D. Szafir, "Robot teleoperation with augmented reality virtual surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 202–210.

[11] A. Bonci, P. D. Cen Cheng, M. Indri, G. Nabissi, and F. Sibona, "Human-robot perception in industrial environments: A survey," *Sensors*, vol. 21, no. 5, p. 1571, 2021.

[12] T. S. Kuthambalayan and S. Bera, "A review of the literature on mixed make-to-stock/make-to-order production systems: major findings and directions for future research," *International Journal of Services and Operations Management*, vol. 37, no. 3, pp. 372–406, 2020.

[13] M. Indri, L. Lachello, I. Lazzero, F. Sibona, and S. Trapani, "Smart sensors applications for a new paradigm of a production line," *Sensors*, vol. 19, no. 3, p. 650, 2019.

[14] V. Annem, P. Rajendran, S. Thakar, and S. K. Gupta, "Towards remote teleoperation of a semi-autonomous mobile manipulator system in machine tending tasks," in *International Manufacturing Science and Engineering Conference*, vol. 58745. American Society of Mechanical Engineers, 2019, p. V001T02A027.

[15] T. Stoyanov, R. Krug, A. Kiselev, D. Sun, and A. Loutfi, "Assisted telemanipulation: A stack-of-tasks approach to remote manipulator control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[16] A. Singh, S. H. Seo, Y. Hashish, M. Nakane, J. E. Young, and A. Bunt, "An interface for remote robotic manipulator control that reduces task load and fatigue," in *2013 IEEE RO-MAN*. IEEE, 2013, pp. 738–743.

[17] Y. Tsukada and T. Hoshino, "Layered touch panel: the input device with two touch panel layers," in *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, 2002, pp. 584–585.

[18] "Robot operating system website," Available online: http://wiki.ros.org.

[19] "Leap motion developer website," Available online: https://developer-archive.leapmotion.com/documentation/v2/cpp/index.html (accessed May 2021).

[20] "Webcam c210 specifications," Available online: https://support.logi.com/hc/en-au/articles/360023462133-C210-Technical-Specifications (accessed May 2021).

[21] "find-object ROS package," Available online: https://github.com/introlab/find-object (accessed May 2021).

[22] M. Labbé, "Find-Object interface," Available online: http://introlab.github.io/find-object (accessed May 2021).

[23] "Opencv website," Available online: https://opencv.org (accessed May 2021).

[24] "Gazebo website," Available online: http://gazebosim.org (accessed May 2021).

[25] NIRYO, "Niryo one mechanical specifications," 2018.

[26] "Raspberry pi website," Available online: https://www.raspberrypi.org/ (accessed May 2021).

[27] "Experimental test video demo," Available online: https://youtu.be/7HvFw0sa3Lg.