VOLUME 17, N° 3 2023
Journal of Automation, Mobile Robotics and Intelligent Systems

MULTIMODAL ROBOT PROGRAMMING INTERFACE BASED ON RGB-D PERCEPTION AND NEURAL SCENE UNDERSTANDING MODULES

Submitted: 14th January 2023; accepted: 24th May 2023

Bartłomiej Kulecki

DOI: 10.14313/JAMRIS/3-2023/20

Abstract:
In this paper, we propose a system for natural and intuitive interaction with the robot. Its purpose is to allow a person with no specialized knowledge or training in robot programming to program a robotic arm. We utilize data from the RGB-D camera to segment the scene and detect objects. We also estimate the configuration of the operator's hand and the position of the visual marker to determine the intentions of the operator and the actions of the robot. To this end, we utilize trained neural networks and operations on the input point clouds. Also, voice commands are used to define or trigger the execution of the motion. Finally, we performed a set of experiments to show the properties of the proposed system.

Keywords: *Human-robot interface, Robot programming, 3D perception*

1. Introduction

Cooperative robots are becoming increasingly popular in modern industry, because they support the production process. Once programmed, they perform repetitive tasks for days or months. Reprogramming a robot is a process that requires robotics expertise. New methods of robot programming are constantly being developed to enable operators to conveniently program robots to perform complex tasks. This article addresses the problem of intuitive robot programming using natural language. The proposed interface does not rely on expensive devices for robot programming, such as operator panels (teach pendants) or virtual reality goggles. The main goal of the integrated system is to interpret the operator's intentions based on camera images and voice commands to program the robot's motions.

Convolutional Neural Networks (CNN) enable the detection of objects in the scene and the interpretation of context related to the robot operator's intentions. To make robot programming more flexible and enable fast and natural programming of movements, in this article, we propose to use 3D perception and CNNs to interpret the operator's intentions. The goal of the designed interface is to mimic human-to-human communication, which is based on words and gestures. The operator uses his hands to point at an object to be grasped or asks the robot to pass an object to his hand (Fig. 1) by giving an appropriate voice command.

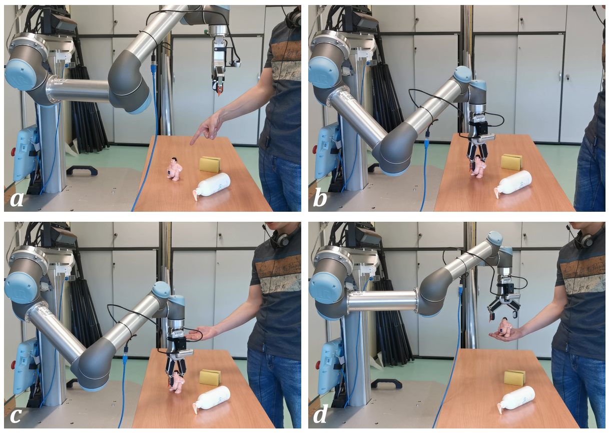

Figure 1. Interaction of the operator with the robot using gestures: pointing at the object by the operator (a), grasping the object by the robot initiated by a voice command (b), passing to hand order (c), transferring an object to the operator's hand (d)

2. Related Work

This article proposes to use various interfaces to define the robot's expected motion. The recognized user's commands can cause scheduling or direct execution of repetitive tasks like in [5, 11]. In this work, we combined hand gesture recognition, voice commands, and scene understanding to implement a new natural human-robot interface for programming the manipulator. Many human-robot interfaces use multiple channels of communication with the user. The popular approach is connecting gestures with voice commands as in [18]. Such a solution prevents many faulty actions caused by the user's mistakes in gestures or misunderstandings by a robot.

One of the most natural ways to interact with a robot is through verbal communication. Xiaoling et al. [17] present a system that controls a mobile robot using voice commands. MFCCs (Mel Frequency Cepstral Coefficients) are extracted as audio signal features. Authors perform speech recognition based on a pattern matching algorithm – Dynamic Time Warping (DTW). A similar and common approach

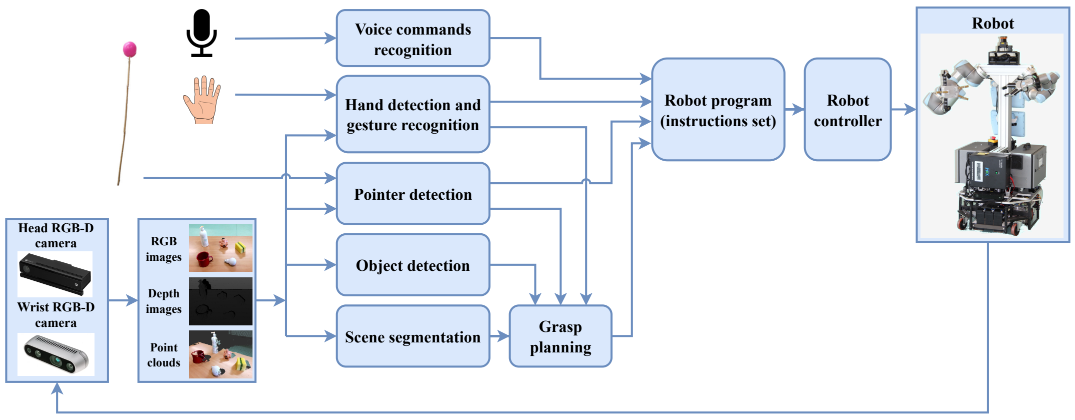

Figure 2. Architecture of the proposed robot programming system – a user can interact with the robot through voice commands, hand gestures, and a pointer to indicate 3D positions and trigger desired actions

is the use of the Hidden Markov Model (HMM). A common solution to the speech recognition problem is to use off-the-shelf software for this purpose, such as Microsoft or Google Speech API [18, 22], CMU Sphinx [23], or wit.ai environment. For the correct performance of the human-robot interface, the voice recognizer module has to convert speech to text, find a keyword in the command, connect it with the specific action, and send an order to the robot. Voice commands greatly simplify the communication between the operator and the robot [1].

Many human-robot interfaces use vision as a source of information for planning the robot's motion. Gesture recognition is an important function of many user interfaces, as it is natural for people to complete a verbal message with a gesture. This method enables the user to select an object in the scene or to show the desired robot position. In [3], the authors implemented an interface to control a dual-arm robot. They used a Leap Motion sensor and estimated the position and orientation of the operator's hands using Kalman and particle filters. The method presented in [4] uses RGB-D images to detect hand gestures and track whole body parts, allowing the operator to select objects on the floor with his arm. Many methods use neural networks to detect hand (or body) positions and configurations and classify gestures [8, 19, 20, 24]. The recognized gestures are then translated into commands corresponding to the defined robot motions. In [6, 7] human action prediction in human-object interactions is performed using a probabilistic framework. Similar to our approach, RGB-D images are utilized.

The operator can use gestures to specify an object for the robot to grasp. Other ways of selecting objects utilize visual markers. A popular approach is based on a laser pointer and detecting the laser spot in the RGB image. This solution is often used in robotic arms supporting partially paralyzed people [9]. A simple detection process and fast selection of objects from a long distance are the main advantages of using a laser pointer. However, this method can be problematic for detecting the laser spot on transparent or reflective objects.

The human-robot interface plays an important role, but only the visual understanding of the scene enables the robot to interact with the *a priori* unknown and ever-changing environment [1, 21]. Object recognition is another field that gives robots new advanced capabilities. Today's state-of-the-art methods are based on Convolutional Neural Networks. Examples include YOLO [2], SSD [16], and Mask R-CNN [10]. Knowledge about the categories of the objects on the scene often helps in grasp planning [14] and enables selecting an object by name.

All robotic systems with user interfaces have to meet the imposed requirements regarding the correct understanding of commands, effectiveness, and time to complete the requested tasks. HRI verification scenarios include the execution of tasks like assembling [18] or objects grasping [4]. One of the most common criteria of verification metrics is a task success rate and execution time [3]. For the system based on voice commands [18] the authors evaluated the time and accuracy of a voice command understanding.

In this research, we used the latest advances in machine learning for hand detection, object detection, and grasping to achieve an efficient and intuitive human-robot interface. The presented results are an extension of the approaches shown in previous papers [11–13]. Compared to our previous publications, this algorithm improves the pointer-less programming presented in [13] by enabling an indication of the target with the finger. Also, voice command communication is developed to work in two-way asynchronous mode. In this work, we integrate vision-based pointer [11] and gesture recognition [13] approaches and perform extended experimental verification of the resulting system.

Our contribution with respect to state of the art can be summarized as follows:
- intuitive robot programming interface based on natural language,
- object selection method based on neural object detector, scene segmentation, and hand gestures,
- a complete architecture of the user interface that improves the robot-human interaction process.

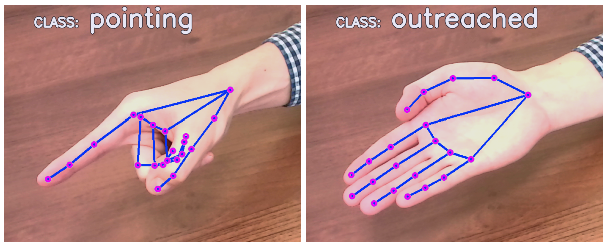

Figure 3. Gestures recognized by the classifier: pointing and outreached

3. Natural Human-Robot Interaction System

This research was conducted using our mobile-manipulating platform Robot 4.0 [14]. During experiments, we do not use the lower part of the robot, but only the Universal Robots UR5 robotic arm presented in Figure 1. The robot is equipped with an Intel Realsense D435 camera mounted on its wrist, and the Microsoft Kinect Xbox One mounted on its head for 3D perception. For grasping, we use an industrial two-fingered OnRobot RG6 gripper. Because of force feedback, using such grippers increase the success rate of many grasp-planning algorithms. The Robot Operating System (ROS) coordinates the functionality of all modules. The robot workspace consists of a table with various objects on its surface.

3.1. System Architecture

The architecture of the proposed system is presented in Figure 2. The user communicates with the robot through voice commands, gestures, and the pointer. Hand, objects, and pointer detection are performed based on RGB-D images. The scene segmentation module processes the point cloud from the camera. After recognizing proper voice commands, estimated hand configuration and pointer position, as well as the information about the objects on the scene, determine the robot's motion.

3.2. Voice Commands

The proposed human-robot interface uses voice commands to confirm intentions expressed by gestures or to command the robot directly. For this purpose, the module that converts speech to text is needed. We decided to use an off-the-shelf solution for this purpose – the SpeechRecognition library and the Google Speech Recognition module, which is easy to integrate, highly effective, and has a choice of many different languages. The beginning of the operator's speech is automatically detected (the module works asynchronously), and the voice command is recorded and sent to the recognizer server (Internet connection is required). After successful speech processing, the user can ensure that the recognized command is valid since the text is converted back to audio format and played.

The set of available commands and associated actions are presented below. We defined each of them in English and Polish with a few equivalent versions which contain synonyms:

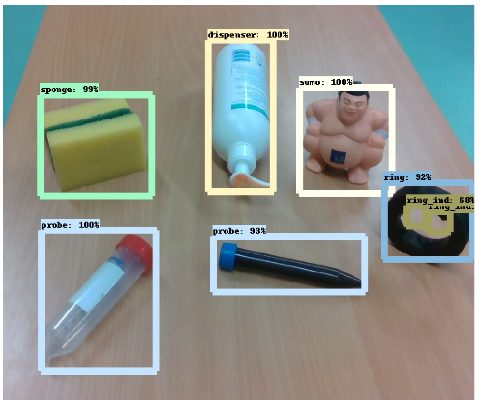

Figure 4. Example output from the object detector – an image with labeled objects

1) ***grasp (+object name):*** the robot plans and executes the trajectory to grasp the selected object and closes the gripper,
2) ***give to hand:*** the robot passes the grasped object to the operator's hand,
3) ***put down:*** the robot places the grasped object in the indicated (or saved) position on the table,
4) ***go there:*** the robot arm (end effector) moves to the position of the detected pointer or hand,
5) ***move home:*** the robot arm moves to the predefined "home" configuration,
6) ***move place:*** the robot arm moves to the predefined "place" configuration,
7) ***go to initial configuration:*** the robot arm moves to the initial configuration,
8) ***look down/front:*** sets end effector pitch angle,
9) ***save initial configuration:*** saves current robot configuration as initial,
10) ***save robot configuration:*** saves current robot configuration as a waypoint,
11) ***save put down position:*** saves indicated position as a place position,
12) ***save trajectory:*** saves current waypoint list as a trajectory,
13) ***start scanning:*** the robot starts scanning the environment (the table and objects),
14) ***open/close the gripper:*** the robot's gripper opens or closes,
15) ***clear/execute program:*** the motion program is cleared or executed,
16) ***play:*** enables robotic arm external control program,
17) ***stop:*** stops the execution of current trajectory.

3.3. Hand Detection and Gestures Recognition

The hand detection in the RGB image is performed using the state-of-the-art system for hand detection and tracking – *Mediapipe Hands* [26]. The output from this module is a set of 21 hand landmark points consisting of 3 coordinates: x, y, and relative depth. Obtained data, which informs us about the hand pose (configuration of hand and fingers), can

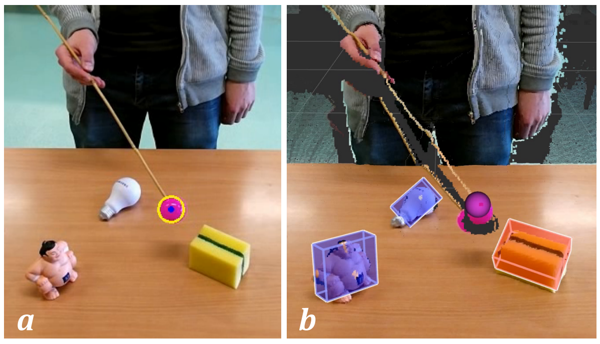

Figure 5. Example pointer detection on RGB image (a) and point cloud with object bounding boxes where red box refers to the selected object (b)

be divided into categories representing gestures. We propose a method that recognizes gestures to allow effective interaction between the user and the robot. Two gestures shown in Figure 3, named *pointing* and *outreached*, are defined. The set of 63 coordinates obtained from the hand detector is used as a feature vector for classification. First, we collected the training data set consisting of 9700 samples (5500 for class *pointing* and 4200 for the *outreached* class). Four different classifiers were trained and tested using the scikit-learn library: Random Forest, Support Vector Classifier, Ridge Classifier, and K-Nearest Neighbors. The final system utilizes the K-NN algorithm that has 99% accuracy on the test dataset containing 2400 samples. The recognized gesture type defines the robot's behavior. If the predicted class is *pointing*, the program searches for the selected object and estimates the indicated point on the table. Conversely, when the result of the classifier is an *outreached* gesture, a target position for the gripper to pass an object is defined on the hand surface.

3.4. Pointer Detection

The main task of the pointer detection module is to find a pointer in the RGB image and estimate its position in 3D space. Later, the obtained 3D coordinates are used to define the target position of the robot end effector, select an object for grasping, or define the place on the table to put the gripped object down. First, the algorithm finds a pointer in the HSV image based on color and shape. Then, a mask is created, which extracts image pixels fitting defined ranges of HSV values corresponding to the color of the pointer. The resulting image is converted to grayscale, thresholded, and blurred using a Gaussian filter. We use the Hough transform to find the circle whose center is the position (x_{px}, y_{px}) of the pointer in the image, as illustrated in Figure 5a. The pointer 3D position is determined using the coordinates of the found circle center, the depth value in the corresponding point in the depth image, and camera intrinsics.

3.5. Object Detection

Our system allows the user to specify a target object without using hands or a pointer. It is possible by including the object name in the voice command. To enable this feature, we implemented an object detection procedure using a Single Shot Detector (SSD) based on the Inception V2 neural network architecture [16]. The SSD was pre-trained on the MS COCO dataset [15]. To create a training dataset, we compiled a set of 10 objects including a sumo figurine, a probe, a dispenser, and a sponge. We collected and manually labeled 1500 images of these objects with bounding boxes and object classes. An example result of the neural network's inference is shown in Figure 4.

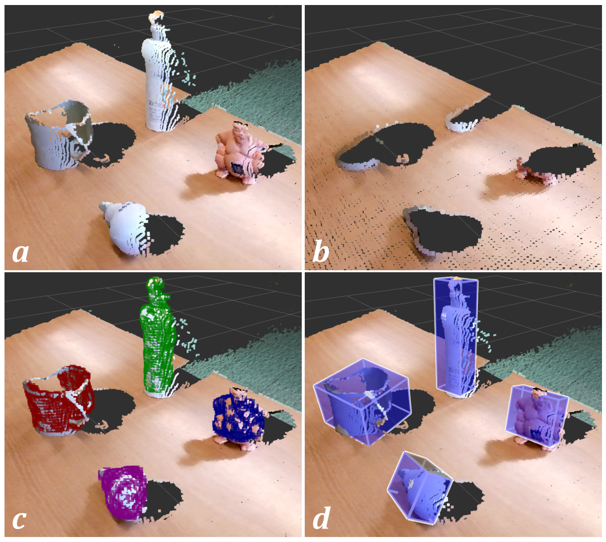

Figure 6. Stages of example scene segmentation process: an input point cloud – from one view (a), estimated table planar (b), point clouds of segmented objects (c), oriented bounding boxes aligned with the input point cloud (d)

3.6. Scene Segmentation

In the proposed system, the 3D perception module performs scene segmentation by processing a point cloud. The output from this module is a set of cuboids that approximate detected objects. The found Oriented Bounding Boxes (OBB) represent the objects' poses and are later used for grasp planning. In the first stage of processing, we detect the table plane using the RANSAC algorithm – by matching the equation of the plane to the data in the point cloud. The resulting plane (Fig. 6b) is removed from the input data to extract points corresponding to the objects. Since the data obtained from a single perspective does not contain information about the complete three-dimensional shape of the object, we perform a scanning procedure to collect and assemble multiple views of the scene. Another solution used in the system is a 3D reconstruction [25], which completes the object point cloud based on the predicted view from its opposite side.

The next step of processing is Euclidean segmentation (based on the Euclidean distance between points) which finds instances of objects in the complete point cloud (Fig. 6c). For each cluster representing an object, the position is estimated by calculating the centroid. Then we perform Oriented Bounding Box fitting to approximate each object. For this purpose, the Principal Component Analysis (PCA) is utilized.

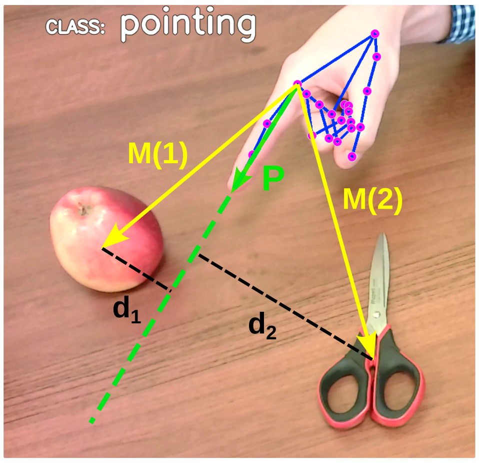

Figure 7. Identification of the pointed object. The smallest distance value d_i is associated with the selected object

It finds the three directions of the object point cloud with the highest variance, which determines the OBB orientation. Example bounding boxes matched to the objects' point cloud are shown in Figure 6d.

3.7. Object Grasping

The grasp-planning method utilizes information about the obtained OBB. The centroid of the object is used as a gripper target position. Orientation of the end-effector depends on the box dimensions and alignment. If the height of the object's OBB is smaller than the width and length, the grip is performed vertically from the top. If the box height is the largest dimension, the robot tries to grasp the object horizontally from the side. The distance between the end-effector and the TCP depends on the gripper opening, which we estimate based on the OBB dimensions. While closing the gripper on the object, force feedback is used. The gripper stops the motion when the reaction force exceeds the threshold value. This strategy compensates for the inaccuracies of the perception system, stabilizes the grasp, and increases the success rate of the grasping method.

3.8. Object Selection

In case of multiple objects on the scene, the user needs a method for specifying the item to grasp. Our system offers three options. The first method uses information about the current gesture and pose of the hand. To determine which object is pointed at by the user's finger, we utilize the position of two hand landmark points detected in the image [26], named INDEX_FINGER_MCP and INDEX_FINGER_TIP (the origin and tip of the index finger). Based on the depth image and the camera matrix, the position of these points in 3D space is obtained, which allows the determination of the pointing vector P. In the next step, the origin of the index finger (MCP point) is connected with the object centers to obtain a set of M vectors. As illustrated in Figure 7, to determine which object is selected by the operator, it is necessary to find the identifier *id* of the OBB that is at a minimum distance (less than the threshold value)

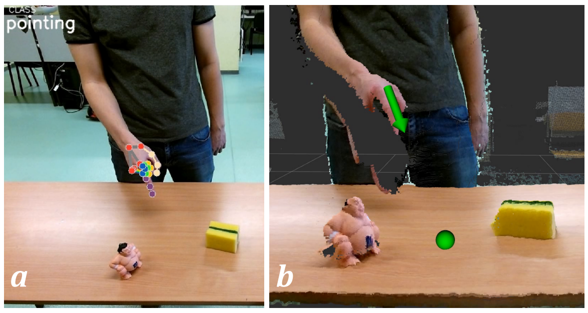

Figure 8. Pointing gesture identified on the RGB image (a) and point cloud visualization (b) with the indicated point on the table (green sphere – point estimated as a line with plane intersection)

from the pointing vector:

$$\underset{id}{\arg\min} \frac{\left\| \vec{P} \times \overrightarrow{M(id)} \right\|}{\left\| \vec{P} \right\|}. \tag{1}$$

Another way to choose an object for grasping is a pointer (Fig. 5). This method calculates the Euclidean distances between the 3D pointer position and centroids of objects on the table. The selected object is chosen only if the smallest distance is below the required threshold value.

Also, the user can specify the target object by voice command. When the output of the neural object detector includes an object with the given name, we find its bounding box in the image. The frame center is converted to 3D space and compared to every object centroid to find the closest (selected) one.

3.9. Calculation of the Goal Position

The set of possible user actions also includes the computation of the goal position of the object. We can interpret the goal as the target position for the end effector or the grabbed object – it depends on the user's voice command. To specify the desired position of the object, the operator has two options. The first method uses the pointer. In this case, the system utilizes the current pointer location to set the position on the table to place the object or the point in space to move the end effector.

Another more intuitive way uses a set of gestures. As mentioned in section 3.3, the robot can pass the grabbed object to the operator's hand when the recognized gesture is "outreached". Also, connected with another voice command, this gesture causes the end effector to move to the hand position. Apart from giving the item to the hand, it is also possible to indicate with a finger the place to place it on the table as illustrated in Figure 8. For this purpose, we determine the intersection of the index finger's straight line given by the point $\vec{W}$ and vector $\vec{P}$ with the plane of the table given by the normal vector $\vec{N}$ and the point $\vec{T}$.

In general, the task is to solve a system of equations:

$$\begin{cases} x = P_x \cdot t + W_x \\ y = P_y \cdot t + W_y \\ z = P_z \cdot t + W_z \\ A(x - x_0) + B(y - y_0) + C(z - z_0) = 0 \end{cases} \tag{2}$$

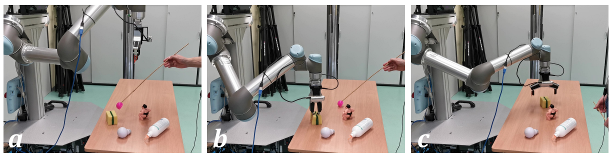

Figure 9. Programming the robot using a pointer: user selects a sponge with a pointer (a), the object is grasped and the user chooses the goal position using a pointer (b), the object is moved (c)

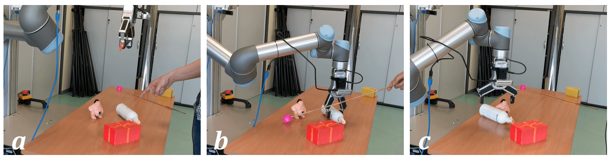

Figure 10. Programming the robot using a mixed approach: user selects a bottle with a gesture (a), the object is grasped and the user chooses the goal position using a pointer (b), the object is moved (c)

where t is a scalar parameter of 3D line equations (parametric form, first three equations in eq. (2)), and

$\vec{P} = [P_x\ P_y\ P_z]^T$	– a vector lying on a line,
$\vec{W} = [W_x\ W_y\ W_z]^T$	– a point belonging to a line,
$\vec{N} = [A\ B\ C]^T$	– a plane normal vector,
$\vec{T} = [x_0\ y_0\ z_0]^T$	– a point belonging to a plane,
$[x\ y\ z]^T$	– a searched line with plane intersection point.

In our case, the task is simplified. The reference frame is placed on the robot's footprint. A table plane normal vector $\vec{N}$ is $[0\ 0\ 1]$ and a point belonging to the table plane $\vec{T}$ can be $[0\ 0\ h_t]$, where h_t is a table height (we estimate it from the point cloud). The $\vec{P}$ vector is calculated as a difference between position vectors of finger TIP and MCP points. The $\vec{W}$ vector is a TIP or MCP point. The last equation simplifies to:

$$z = h_t \tag{3}$$

so we can determine t as:

$$t = \frac{h_t - W_z}{P_z} \tag{4}$$

and substituting to the first two equations calculate x and y – the coordinates of the intersection point:

$$\begin{cases} x = P_x \cdot \frac{h_t - W_z}{P_z} + W_x \\ y = P_y \cdot \frac{h_t - W_z}{P_z} + W_y \\ z = h_t \end{cases} \tag{5}$$

3.10. Robot Program and Motion Planning

The goal of the previously described modules is to define the robot's motions. For motion planning, we use the MoveIt package. The robot executes joint-space or Cartesian-space trajectories depending on the task requirements, for example, the last motion before grasping has to be linear. The MoveIt package is also used to prevent the robot from self-collisions and from collisions with the table (we represent the obstacles with Octomap).

During programming, the motions are executed on the fly and stored in memory. If any system component returns an error, the robot does not perform any action, and the command has to be repeated correctly. The user is supported by a visualization on the screen that allows him to validate the issued commands and gestures. After the operator says the *execute program* command, saved motions can be read and used in repetitive tasks. When the operator wants to program a new motion sequence, he can delete the current program with the *clear program* command. A defined robot program consists of a list of movements and high-level tasks, e.g., moving to the initial pose or grasping an object of a given category.

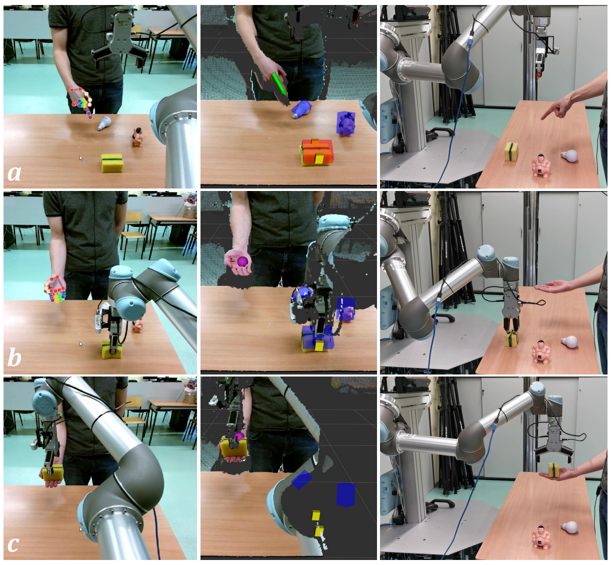

Figure 11. Example robot camera image (left column), point cloud visualization (middle column), and side view (right column) during the grasping experiment: user selects the object with his finger (a), the object is grasped (b), the object is passed to detected hand (c)

4. Results

The capabilities of the proposed robot programming interface were tested in experiments divided into two groups. The first group of experiments was designed to check the time of programming motion sequences of the robotic arm, in which the robot performs simple motions and gripper operations (voice command sequences 1–3). For these trajectories, we compared the proposed intuitive interface with the standard programming methods using the operator panel (teach pendant) and programming by moving the robot manually (teaching by doing – TbD). The goal of experiments in the second group was to review the performance of the user interaction modules during programming higher-level tasks. These tasks include picking and placing a selected object in the specified position on the work table, as well as

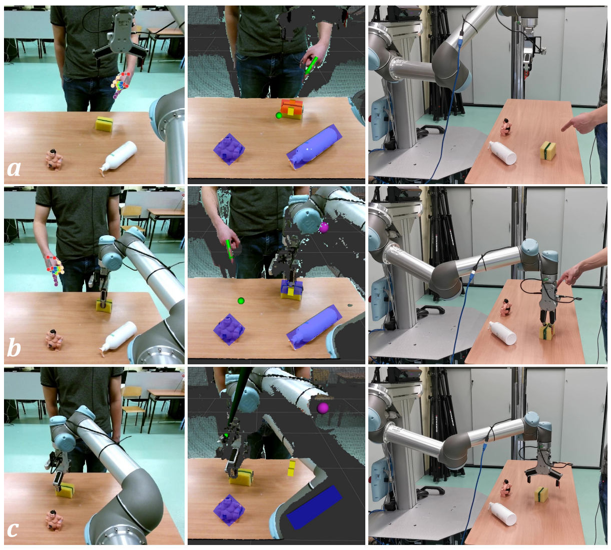

Figure 12. Example robot camera image (left column), point cloud visualization (middle column), and side view (right column) during the grasping experiment: user selects the object with his finger (a), the object is grasped (b), the object is released in the place indicated with a finger (c)

grasping and passing an item to the user's hand (voice command sequences 4–8).

In this paper, we repeat the experiments performed in previous works [11–13] to test the robot programming interface efficiency after making changes in the speech recognition module. We also added another test case that validates the indicating place on the table with a finger.

The following voice command sequences were used during tests:

1–3. various length combination of commands: *save initial configuration*, *go there*, *open/close the gripper*, *save robot configuration*, *save trajectory*, *execute program*

4. *grasp* (with pointer), *put down* (with pointer), *go home*
5. *grasp + name* (with object detection), *put down*, *go home*
6. *grasp* (with gesture), *give to hand*, *go home*
7. *grasp* (with gesture), *put down* (with gesture), *go home*
8. *grasp* (with gesture), *put down* (with pointer), *go home*

During the experiments, we measured the programming time of each trajectory for different programming techniques: the standard methods and using voice commands in combination with a pointer, object detection, or gestures. The results are shown in Table 1. For safety reasons, the robot's speed during the tests was limited to 50%.

Compared to our previous works [11–13] time of programming in each of the proposed methods has decreased. The main reason is the recognition of voice commands in the user's native language. During the experiments, we got the following results: 355/500 (71%) correct conversions in English, and 479/500 (96%) in Polish (native). For successful speech-to-text processing, the orders have to be pronounced clearly and with the correct accent. Speaking in the native language significantly increases the effectiveness of speech recognition, reduces the need to repeat commands, and thus shortens the robot programming time. In the utilized module, the language is easily configurable and can be changed even when the system is running. The efficiency of verbal communication with the robot is also improved by audio feedback. After speech-to-text processing, the output text is converted back to audio and played. It ensures that the user's command has been correctly recognized and informs him when the order is accepted. However, speech recognition has a significant impact on extending programming time. The mean time of recording, converting to text, and playing back the command during our experiments is about 6 seconds. This time strongly depends on the length of the command, the recorded audio signal quality, and the internet connection.

The efficiency of executing the programmed manipulation task heavily depends on the grasping algorithm. The point-cloud-based method used in our system [14] is not able to plan a successful grasp for small objects or those with complicated geometry. For such items (plastic shaft, ring lying down) the success rate is 7.5%. Although, the algorithm performs well for many other objects (e.g. sponge, dispenser bottle, sumo figurine, tape, ring (standing vertically), and I-shape part). For this set of objects, the success rate is 76%. Another important metric that characterizes our system is the accuracy of object recognition. The SSD object detector achieves 76.9% mAP [16].

As we can see in Table 1, programming simple trajectories with a pointer and voice commands can be faster than standard programming methods with the teaching pendant but is still not as fast as the teaching by doing method. The advantage of the proposed interface is the increased intuitiveness and possibility to program higher-level tasks. The user can use the pointer to select the object to pick and indicate its place position, as illustrated in Figure 9.

Example experiments verifying interaction with the robot using only gestures are shown in Figures 11 and 12. In this test case, the robot passes the selected object to the user's hand (command sequence no. 6) or places chosen item in the indicated position (command sequence no. 7). Based on the results shown in Table 1, it can be seen that programming the robot using gestures is slightly more time-consuming than using the pointer. Detecting the hand and calculating its position takes an average of 56 ms (for the pointer it is 36 ms [11]). The main advantage of this type of interaction with the robot is that there is no need to use additional hardware (pointer, marker). When programming the robot using the visualization on the computer screen, the user receives feedback on the current interpretation of his gestures and voice commands, which offsets the inconvenience associated with the delay in the system.

Table 1. Comparison of programming time [s] of example motion sequences using six programming methods. The symbols μ and σ denote mean and standard deviation values calculated for 10 measurements

	sequence no.	1		2		3		4		5		6		7		8	
	number of commands	10		12		16		3		3		3		3		3	
	measurement	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
programming time [s]	teach pendant	162.0	16.7	184.3	15.3	235.3	15.9	–	–	–	–	–	–	–	–	–	–
	TbD	**100.0**	**12.3**	**113.4**	**9.9**	**155.4**	**12.5**	–	–	–	–	–	–	–	–	–	–
	pointer and voice cmd. [11]	133.2	7.3	141.8	8.0	188.8	8.8	59.3	1.6	–	–	–	–	–	–	–	–
	object detection and voice cmd. [11]	–	–	–	–	–	–	–	–	68.4	4.4	–	–	–	–	–	–
	gestures and voice cmd. [13]	143.2	7.8	151.5	8.9	206.5	9.3	–	–	–	–	62.0	4.5	63.8	5.9	–	–
	pointer, object detection, gestures and voice cmd.	133.2	7.3	141.8	8.0	188.8	8.8	**59.3**	**1.6**	**68.4**	**4.4**	**62.0**	**4.5**	**63.8**	**5.9**	**73.6**	**5.7**

During the tests, the performance of the interface integrating the different interaction modules was also checked. We performed an experiment illustrated in Figure 10, in which the user selects an object to grasp using a gesture and then uses a pointer to determine the place on the table where the object should be put down (commands sequence no. 8). We observe that the system using multiple modules for human-robot interaction at the same time runs slower because it requires more computing resources. However, this type of solution offers the greatest flexibility. For example, it allows the selection of an object of any type, unlike a system with object detection, which only lets the user choose items that were in the training set. Only the integrated system presented in the article allows all defined tasks to be programmed.

5. Conclusion

This article proposes a system that uses voice commands, gestures, and 3D perception to program a robot to perform complex tasks. The advantage of the designed system is the capability to easily program tasks related to grasping and moving objects, which is not possible with standard methods (teach pendant, teaching by doing). In addition, an increase in the intuitiveness of the robot's operation was achieved through the use of gesture recognition combined with voice commands. The integrated robot interaction modules allow the robot to be programmed not only with gestures but also with a pointer. In the future, we plan to increase the number of recognized gestures to better understand the user's intentions and improve human-robot interaction.

AUTHOR
Bartłomiej Kulecki – Poznan University of Technology UL. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: bartlomiej.kulecki@put.poznan.pl, www.put.poznan.pl.

ACKNOWLEDGEMENTS
This work was supported by the National Centre for Research and Development (NCBR) through project LIDER/33/0176/L-8/16/NCBR/2017.

References

[1] R. Adamini, N. Antonini, A. Borboni, S. Medici, C. Nuzzi, R. Pagani, A. Pezzaioli, and C. Tonola. "User-friendly human-robot interaction based on voice commands and visual systems," *2021 24th International Conference on Mechatronics Technology* (ICMT), 2021, pp. 1–5.

[2] A. Bochkovskiy, C. Wang, and H.M. Liao. "YOLOv4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[3] M. Chen, C. Liu, and G. Du. "A human–robot interface for mobile manipulator," *Intelligent Service Robotics*, vol. 11, no. 3, 2018, pp. 269–278; doi: 10.1007/s11370-018-0251-3.

[4] P. de la Puente, D. Fischinger, M. Bajones, D. Wolf, and M. Vincze. "Grasping objects from the floor in assistive robotics: Real world implications and lessons learned," *IEEE Access*, vol. 7, 2019, pp. 123725–123735.

[5] W. Dudek and T. Winiarski. "Scheduling of a robot's tasks with the tasker framework," *IEEE Access*, vol. 8, 2020, pp. 161449–161471.

[6] V. Dutta, and T. Zielińska. "Predicting human actions taking into account object affordances," *Journal of Intelligent & Robotic Systems*, vol. 93, 2019, pp. 745–761.

[7] V. Dutta, and T. Zielińska. "Prognosing human activity using actions forecast and structured database," *IEEE Access*, vol. 8, 2020, pp. 6098–6116.

[8] Q. Gao, J. Liu, Z. Ju, and X. Zhang. "Dual-hand detection for human–robot interaction by a parallel network based on hand detection and body pose estimation," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, 2019, pp. 9663–9672.

[9] M. Gualtieri, J. Kuczynski, A.M. Shultz, A. Ten Pas, R. Platt, and H. Yanco. "Open world assistive grasping using laser selection," *2017 IEEE International Conference on Robotics and Automation* (ICRA), 2017, pp. 4052–4057.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN," *2017 IEEE International Conference on Computer Vision* (ICCV), 2017, pp. 2980–2988.

[11] B. Kulecki. "Intuitive robot programming and interaction using RGB-D perception and CNN-based objects detection," *Automation 2022: New Solutions and Technologies for Automation, Robotics and Measurement Techniques*, R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds., Cham, 2022, pp. 233–243.

[12] B. Kulecki, and D. Belter. "Intuicyjny interfejs programowania robota z neuronowymi modułami do interpretacji sceny," *Postępy robotyki. T.2*, C. Zieliński and A. Mazur, eds., Warszawa, 2022, pp. 89–98.

[13] B. Kulecki, and D. Belter. "Robot programming interface with a neural scene understanding system," *Proceedings of the 3rd Polish Conference on Artificial Intelligence, April 25–27, 2022, Gdynia, Poland*, 2022, pp. 130–133.

[14] B. Kulecki, K. Młodzikowski, R. Staszak, and D. Belter. "Practical aspects of detection and grasping objects by a mobile manipulating robot," *Industrial Robot*, vol. 48, no. 5, 2021, pp. 688–699.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. "Microsoft COCO: Common objects in context," *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds., Cham, 2014, pp. 740–755.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A.C. Berg. "SSD: Single shot multibox detector," *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., Cham, 2016, pp. 21–37.

[17] X. Lv, M. Zhang, and H. Li. "Robot control based on voice command," *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 2490–2494.

[18] I. Maurtua, I. Fernández, A. Tellaeche, J. Kildal, L. Susperregi, A. Ibarguren, and B. Sierra. "Natural multimodal communication for human–robot collaboration," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, 2017; doi: 10.1177/1729881417716043.

[19] O. Mazhar, S. Ramdani, B. Navarro, R. Passama, and A. Cherubini. "Towards real-time physical human-robot interaction using skeleton information and hand gestures," *IEEE/RSJ Int. Conf. on Int. Robots and Systems* (IROS), 2018, pp. 1–6.

[20] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, and G. Sansoni. "Deep learning-based hand gesture recognition for collaborative robots," *IEEE Instrumentation Measurement Magazine*, vol. 22, no. 2, 2019, pp. 44–51.

[21] K.-B. Park, S. H. Choi, J. Y. Lee, Y. Ghasemi, M. Mohammed, and H. Jeong. "Hands-free human–robot interaction using multimodal gestures and deep learning in wearable mixed reality," *IEEE Access*, vol. 9, 2021, pp. 55448–55464.

[22] P. Putthapipat, C. Woralert, and P. Sirinimnuankul. "Speech recognition gateway for home automation on open platform," *2018 International Conference on Electronics, Information, and Communication* (ICEIC), 2018, pp. 1–4.

[23] S. Sharan, T. Q. Nguyen, P. Nauth, and R. Araujo. "Implementation and testing of voice control in a mobile robot for navigation," *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (AIM), 2019, pp. 145–150.

[24] M. Simão, P. Neto, and O. Gibaru. "Natural control of an industrial robot using hand gesture recognition with neural networks," *IECON 2016 – 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 5322–5327.

[25] R. Staszak, B. Kulecki, W. Sempruch, and D. Belter. "What's on the other side? A single-view 3D scene reconstruction," *2022 17th International Conference on Control, Automation, Robotics and Vision* (ICARCV), 2022, pp. 173–180.

[26] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. "Mediapipe hands: On-device real-time hand tracking," *arXiv e-prints*, Jun 2020; doi: 10.48550/arXiv.2006.10214.