Procedia CIRP

A Voice-Based Programming Approach and Framework for Collaborative Robotic Arms

--Manuscript Draft--

Manuscript Number:	PROCIR-D-19-01531R1
Article Type:	SI: CATS 2020
Section/Category:	SI: CATS 2020
Corresponding Author:	Tudor B. Ionescu, Dr.-Ing. Technical University of Vienna Vienna, AUSTRIA
First Author:	Tudor B. Ionescu, Dr.-Ing.
Order of Authors:	Tudor B. Ionescu, Dr.-Ing.
	Sebastian Schlund, Prof. Dr.-Ing.
Abstract:	We present a new voice-based programming approach and software framework for collaborative robots (cobots) based on the Web Speech API, which is now supported by most modern browsers. The framework targets human programmable interfaces (HPIs), which can be used by people with little or no programming experience. The framework follows a meta-programming approach by enabling users program cobots by voice in addition to using a mouse, tablet or keyboard. Upon a voice instruction (such as move, pick, release, etc.), the framework automates the manual tasks required to manipulate the vendor-provided HPI. The main advantages of this approach are simplified, guided programming, which only requires the knowledge of 5–10 voice instructions; increased programming speed by up to 46% compared to the manual approach; and the possibility of sharing programs as videos. The open-source framework was evaluated using two application scenarios.

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

8th CIRP Conference of Assembly Technology and Systems

A Voice-Based Programming Approach and Framework for Collaborative Robotic Arms

Tudor B. Ionescu[a], Sebastian Schlund[a]*

[a]*Vienna Technical University, Theresianumgasse 27, Vienna-1040, Austria*

* Tudor B. Ionescu – *E-mail address:* tudor.ionescu@tuwien.ac.at

Abstract

We present a new voice-based programming approach and software framework for collaborative robots (cobots) based on the Web Speech API, which is now a W3C standard component of most modern browsers. The framework targets human-programmable interfaces (HPIs), which can be used by people with little or no programming experience. The framework follows a meta-programming approach by enabling users program cobots by voice in addition to using a mouse, tablet or keyboard. Upon a voice instruction (such as *move, pick, release, etc*,), the framework automates the manual tasks required to manipulate the vendor-provided HPI. The main advantages of this approach are simplified, guided programming, which only requires the knowledge of 5–10 voice instructions; increased programming speed by up to 40% compared to the manual approach; and the possibility of sharing programs as videos. The open-source framework was evaluated using a real assembly application and human participants.

Keywords: Collaborative robots, intuitive programming, voice-based programming

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

8th CIRP Conference of Assembly Technology and Systems

Programming Cobots by Voice: A Human-Centered, Web-Based Approach

Tudor B. Ionescu [a]*, Sebastian Schlund [a]

[a]*Human-Machine Interaction Group, Institute of Management Science, Technische Universität Wien; Theresianumgasse 27, 1040-Vienna, Austria.*

* Corresponding author. Tel.: +43 (1) 58801-33057; fax: +43 (1) 58801-33098. E-mail address: tudor.ionescu@tuwien.ac.at

Abstract

We present a new voice-based programming approach and software framework for collaborative robots (cobots) based on the Web Speech API, which is now supported by most modern browsers. The framework targets human programmable interfaces (HPIs), which can be used by people with little or no programming experience. The framework follows a meta-programming approach by enabling users program cobots by voice in addition to using a mouse, tablet or keyboard. Upon a voice instruction (such as move, pick, release, etc.), the framework automates the manual tasks required to manipulate the vendor-provided HPI. The main advantages of this approach are simplified, guided programming, which only requires the knowledge of 5–10 voice instructions; increased programming speed by up to 46% compared to the manual approach; and the possibility of sharing programs as videos. The open-source framework was evaluated using two application scenarios.

Keywords: Type your keywords here, separated by semicolons ;

1. Introduction

Assembly automation has increasingly developed over the last decades also thanks to the introduction of collaborative robotic arms (short, cobots), which can work in close proximity to humans without requiring a safety fence. Cobots are currently used to automate repetitive manual tasks, such as loading/unloading of machines, pick & place operations, screwing tasks, or quality inspection. Cobots can help to accomplish some of these tasks more cost effectively than humans in productive industrial contexts. Nevertheless, the field of hybrid automation—i.e., where humans and machines work together in direct interaction has not been realized to the extent projected by the industrial automation community and cobot vendors.

Currently, a number of factors sum up to prevent the wide scale adoption of cobots in Europe. First, in most European countries, each application requires its own safety certification, which prevents users to change the application layout and program without conducting a new risk assessment. Second, current robot programming environments are usually considered counter-intuitive because they build upon older expert programming models, such as function blocks, textual programming, and combinations of visual and textual programming modes. At the same time, newer robot programming environments, which draw upon design patterns used for smart phone user interfaces and other commonly used day-to-day technologies, provide opportunities for non-experts to engage with programming cobots more easily [1]. Yet, while the barriers are lowered, these environments often lack the versatility of textual programming and are subjected to app-oriented business models, which foresee the acquisition of additional robot apps (or skills) at prohibitive costs. And third, while robot vendors strive for targeting new markets and users, in the industrial domain, robot programming is still regarded as being a task for experts. This leads to a gap concerning a learner's transition from multi-model, intuitive robot programming towards the more versatile, textual programming environments. Such a transition would, for example, allow assembly workers to program robots using increasingly complex programming models.

In response to this situation, we envision a scenario in which robot programming is certified as an assembly application instead. This would allow for workers to design applications in collaboration with assembly planners without the need for recertification. In this context, the collaboration between humans and robots occurs at the cognitive level, with workers and other non-experts designing and programming applications. Then, depending on lot size, certification costs, and other factors, organizations may choose to operate cobots behind a fence or not. In any case, all employees would benefit from directly programming cobots on the long term, as industrial robot safety research is expected to lower the barriers required to certify industrial cobot applications.

Drawing on the assumption that the notion of intuitive user interfaces is contingent on the user's prior knowledge and experience with technologies having similar user interfaces as well as with analogous day-to-day activities (i.e., conversations, text chatting, web searching, interactions with smart phones and tablets, etc.) we hypothesize that intuitive robot programming is not a holy grail to be attained by one human-programmable interface (HPI); but a matter of providing diverse supports for that which is already known and experienced by the projected users. These supports may include app-oriented drag-and-drop features, voice-based programming, and other multimodal interactions.

In this paper, we dive into voice-based programming as one mode of supporting a more intuitive interaction with cobots. This feeds into the scenario in which cobot programming is safety-certified as an industrial application and workers can accomplish a great number of programming tasks on the shop floor. We begin by discussing cobot programming by voice in the broader context of speech-based programming. Then, we introduce a new concept and lightweight tool for voice-based cobot programming, which leverages existing technologies; and then evaluate it on the basis of two usage scenarios.

1.1. (Robot) Programming by Voice

Programming by voice or speech is a compelling idea, which received some attention by the software engineering and robotics communities in the past 25 years. While during the 2000s the main problem appeared to be the lack of reliable speech recognition techniques, owing to the recent advances is machine learning, today researchers are confronted with other problems, such as how to structure voice commands in order to produce code while minimizing the necessary corrections [8,9]. In this sense, to reduce the speech recognition error, some researchers use code words instead of 1-to-1 mappings between what is being said/recognized and what is being coded. In the industrial robotics domain, programming by voice (i.e., producing program code rather than controlling or guiding a robot by voice) has received relatively little attention compared to other programming techniques, such as block-based [3], automated [4] programming, or multimodal teach-in [5]. More work appears to have been invested in controlling or guiding a robot using voice commands [2], whereby these commands are used to immediately call some pre-programmed function of the robot (e.g., different movements or invocation of more complex robotic skills). In the case of collaborative industrial robots (cobots), "conversations" between humans and robots on the shop floor are often very limited and can sometimes even be annoying, as the buzz of the factory is interfering with the intentions of the user. The advantages of operatively controlling cobots by voice are also limited by the requirement to certify cobot applications individually, which leaves little room for creativity during operation. In this context, simple buttons are often more effective in triggering or acknowledging cobot actions in productive applications.

Against this background, this paper presents a novel lightweight cobot programming by voice framework based on the new Web Speech API (WSAPI)—a W3C specification published and maintained by the Speech API Community Group supported by modern web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge. The framework explicitly targets the assembly application development phase, which usually requires many rounds of repetitive trial and error until robots reach the necessary precision, especially in small part assembly, such as PCB assembly. In this context, programming and testing cobot applications by voice can be more effective in terms of speed and difficulty. This would also empower shop floor employees to participate in the application development activities, as a growing body of work has called for (e.g., [1,6]).

The framework we developed uses the WSAPI to wrap the programming structures used by a cobot produced by the Franka Emika company, which already offers a simplified web-based human programmable interface (HMI). In this environment, non-experts can program the robot by dragging, dropping, and configuring predefined apps in a sequence or according to simple patterns, such as repetitions of the same action or more complex composite apps, which can be acquired in an app store. This app-oriented way of programming cobots appears to be more intuitive than other industrial cobot HPIs because it draws on known patterns of interactions (e.g., with smart phones and tablets). This provides non-programmers with the opportunity of a fast and easy initiation into cobot programming. Nevertheless, the same programming technique becomes rather annoying and slow once a person is initiated into the craft of cobot programming. In this context, programming by voice can help to bridge the potential interest gap, which may arise when a person finds something to be too easy and limited. Also, and perhaps more importantly from an economic point of view, programming cobots by voice can boost the productivity of both beginner and expert programmers by allowing them to use their hands for guiding the robotic arm rather than inputting text or dragging apps and functions. The framework introduced in this paper thus aims to augment the multimodal programming capabilities of existing cobots rather than providing a definitive replacement for any of them. Also, while the framework currently only support the Franka Emika Panda robot, we are already working on a version for the Universal Robot series, which will become available in the course of 2020.

2. Cobot Programming by Voice: A Web Framework

This section provides technical details about the design and implementation of the web-based programming by voice framework discussed in this paper. We begin by introducing the Panda robot's native HPI and the WSAPI before describing the software components needed to extend the capabilities of this interface.

2.1. Programming the Panda Cobot: Features & Limitations

The Franka Emika Panda collaborative, industrial-grade robotic arm is one of the first of its kind to provide a web-based HPI. The design of this HPI closely follows the model-view-controller (MVC) pattern [7], with a REST and Websockets backend and a dynamic HTML5 & JavaScript web interface. The REST service components of Panda's HPI control and maintain the robot's state by processing sensor signals and actuating robot moves. The backend application, which implements the model in the MVC pattern, wraps the robot's native C++ API, called libfranka. This model maintains the current state of the robot (i.e., joint positions, applied forces, sensor values, actuation history, etc.) in memory. The web application, shown in Fig. 1, is updated via web-sockets to reflect the current state of the model. In this web environment, users can create new robot applications by dragging and dropping so-called robot apps from an app collection to a workflow pane to form a sequence of basic actions, whereby each app in the sequence represents a configurable instance of a movement or behavior of the robot (e.g., taught-in or relative moves, gripper actions, waiting for user input, etc.). In addition, there exist so-called pattern apps, which implement predefined applications patterns (e.g., picking and/or placing parts in a tray, or more advanced program structures). Named app sequences, which are called tasks, are persisted in the model, whereby all user interactions generate calls to the REST services.

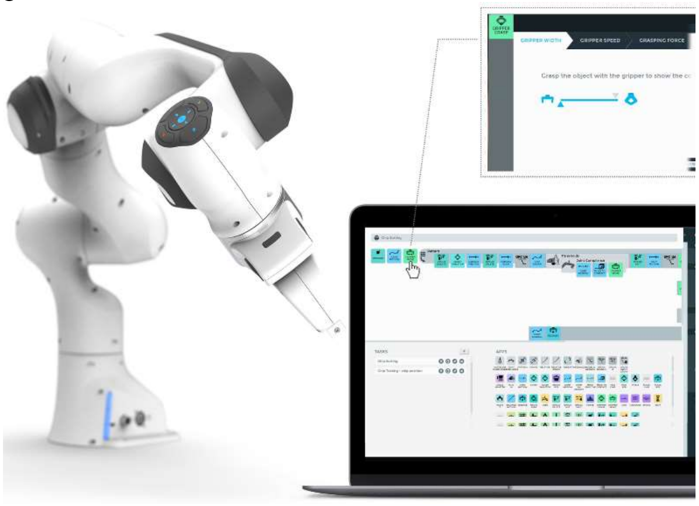

Fig. 1. The Franka Emika Panda robot and HPI.

This app-oriented HPI is coupled with a series of buttons available on the robot's head, as shown in Fig. 1. Two of these buttons are used to move the arm in free-drive mode. The other ones are used within the wizard-like configuration of each app, which is activated by clicking on the respective app, as shown in Fig. 1. This way the user only needs to interact with the laptop or tablet when dragging and dropping a new app into the workflow. For most of the available apps, all other parameters can be configured using the buttons from the robot's head. In practice, however, users prefer to set certain parameters and click through the wizards using the mouse or touch screen of the laptop of tablet because the buttons are relatively hard to press and thus induce physical strains into the wrist when used repeatedly for a long time. Also, configuring the apps using the buttons is usually slower than using the mouse and keyboard. Hence, while Panda's HPI arguably represented a leap in intuitive robot programming, it has some obvious limitations when it comes to ergonomics and speed.

In our experience with using this robot in research and teaching activities, these limitation not only reduce the productivity of users, who need to switch back and forth between the robot's head and the computer, but may also pose a safety hazard when more than one individual works with the robot. Students, for example, tend to divide tasks so that one person will more the robot, while another one configures the apps. At times, however, both persons may stare at the computer screen, whereby one of them (the one who guided the robot's head) may be in close proximity to the robot, which—on occasion—may perform unexpected moves (e.g., in case of a software or hardware error) and thus potentially harm users. As we argued elsewhere, when working in teams of two, only one person should program the robot, while the other one should watch that all safety-related rules are being followed [1]. These roles should be switched periodically.

2.2. Leveraging the Web Speech API to Enable Cobot Programming by Voice

The WSAPI is a collection of client-sided JavaScript functions embedded in modern web browsers (notably Chrome and Firefox), which allows web developers to leverage the newest machine-learning based speech recognition technologies. Common use cases include voice web search, speech command interface, speech translation, dialog systems, multimodal interaction and search, etc. The WSAPI can access the computer's microphone (or any other microphone connected to it) via the browser. To ensure privacy, the user is asked for permission before listening is enabled. As opposed to older speech recognition frameworks, the WSAPI supports a wide variety of languages and has a very high recognition success rate. This makes corrections only necessary in the presence of external acoustic interferences, provided that the user is a fluent speaker in his/her language of choice.

Once enabled, the WSAPI listens to what is being said and converts spoken sequences into words placed, for example, in an array. To implement cobot programming by voice, we developed a script that can be injected into Panda's web-based HPI using the "bookmarklet" technique. A bookmarklet is a bookmark containing JS commands stored in the browser's "favorites" bar, which adds "one-click" functionalities to the currently displayed web page. This technique has been effectively used to embed or extend programming environments in existing webpages (e.g., [20]). The current approach uses the same mechanism to inject JavaScript code into Panda's HPI, which automates the instantiation, loading, configuration, and execution of apps and tasks in the robot's native HPI according to the user's voice commands.

The bookmarklet technique we used to extend Panda's HPI by voice programming falls into the broader category of graphical user interface (GUI) automation, which is primarily used in software testing. Yet, with the increasing diversity and complexity of software used in companies and privately, more GUI automation use cases emerged in the past few years. For example, the award winning UiPath toolset supports the development of so-called software robots that automate repetitive, annoying user interactions with software systems such as SAP or Microsoft Excel. In this spirit, the proposed framework aims to bring this new philosophy of interfacing with existing software GUIs to the world of collaborative robotics programming to the end of boosting productivity.

In the following we provide some technical details about the implementation of the framework before turning to the an empirical evaluation and discussion of the approach.

2.3. Framework Design and Features

Fig. 2 illustrates the architecture of the "Programming by Voice" extension to the Panda robot's HPI. Building on the robot's HPI, which implements the well-known Model-View-Controller (MVC) design pattern, the user injects a so-called "Programming by Voice" (PBV) controller into the current browser page (i.e., the robot's HPI shown in Fig. 1). Once injected, the PBV controller activates and maintains an open connection to the Web Speech API embedded in the browser, which enables it to listen for voice commands coming from the user. Upon recognizing one of the commands listed in Table 1, the PBV controller may use the visual controls available to users by automating the use actions needed to accomplish certain tasks (e.g., to drag, drop, or configure different robot behavior apps). The PBV controller can also directly manipulate the server-side components of the system by sending REST service calls. One the server side, the application model (which contains the robot's state, programmed tasks, and app implementations) is maintained and persisted in memory. The robot's HPI is thus augmented by a new programming mode, which enables users to create tasks while keeping their hands free (e.g., for teaching positions by guiding the robot and configuring the gripper using both hands). This eliminates the need for switching back and forth between the robot and the laptop or tabled running its HPI.

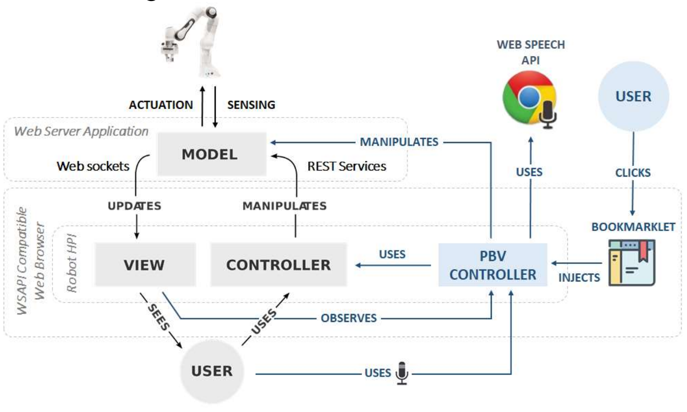

Fig. 2. Architecture of the "Programming by Voice" extension.

The following commands are supported in the current implementation, which can be divided into three distinct groups, depending on the context of use:

Group	Command	Functionality
1	*Speed/Force/ Load/Speed-up <#>*	Sets the velocity / grasping force / gripper load / acceleration of the arm for all subsequent apps to the specified integer value <#>.
2	*Grasp*	Initializes the "Gripper Grasp" app which allows users to manually set the opening of the gripper so as to grasp a certain object. After triggering this app, the users are able to use the buttons on the robot's head or to manually set the gripper fingers to the desired position.
	Hand	Initializes the "Gripper Move" app which allows users to set the opening of the gripper using the buttons on the robot's head or by manually moving the gripper fingers.
	Motion	Initializes the "Cart Motion" app, which allows users to teach in a movement of the arm using several waypoints. Waypoints are set by driving the robot's arm to the desired pose and then pushing the OK button on its head.
3	*Okay*	After having set all the parameters of an app by using the buttons on the robot's head and/or manually moving the robot's arm and fingers, this command will auto-complete all the information requested in the subsequent dialogues of the app-specific wizard (e.g., velocity, force, acceleration, load).
	Start	This command starts the execution of the task being programmed.
	Stop	This command stops the execution of a task.

The first group of commands can be issued at any time while working with the robot's native HPI. These commands save time and clicks by allowing users to set reusable parameters before configuring specific apps. While configuring an app, which requires certain parameters, issuing the "Okay" command will automatically set the values of the app-specific parameters to those of the matching global parameters. If users forget to set these global parameters to the desired value, they can still be changed at a later time using the app's configuration wizard.

The second group of commands instantiate apps for actuating the robot's arm and tool (e.g., gripper). In the robot's native HPI, these off-the-shelf apps need to be dragged from the app library and dropped in the workflow pane of the currently active task. The task of instantiating apps, which can take 5-10 seconds per app, can be automated using voice commands. This saves time because, while the robot's HPI performs the initialization of a task, the user can already drive the robot into the desired pose for the first waypoint of the motion or set the width of the gripper fingers. After all waypoints are configured using the buttons on the robot's head and/or the gripper width is manually set by moving its fingers, the user can issue the "Okay" command to autocomplete the remaining app parameters, such as the velocity and acceleration of a movement and the force and load for pick and place operations.

The third groups of commands is used to trigger actions, which would require one or several mouse clicks or tapping the touch screen of the device running the robot's HPI (e.g., laptop, tabled, etc.). The "Okay" command triggers a context-dependent series of actions so as to autocomplete the parameters of the app being configured and to automatically click-through all remaining dialogues of the respective app configuration wizard. The "Start" and "Stop" commands will simply start and stop the currently active task. By using voice commands the user can focus on interacting with the robot through haptic means. To switch from the task programming into the task testing and running mode, the user needs to pull up the robot's safety stop button, which is pushed down during teaching and programming. Hence, using the start/stop commands, users can save time, for example, by allowing users to focus on the robot's actions during testing while keeping the safety stop button in their hands. This also contributes to a safer interaction with the robot during testing, since the user's attention is not distributed between the robot arm and HPI screen.

3. Evaluation of the Approach

In this section we present an evaluation of the programming by voice concept and extension on the basis of two application scenarios, which assume that shop floor employees (e.g., assembly workers and planners) engage in programming the robot in the following situations:

- **Scenario 1:** Fast programming of pick and place operations for small lot sizes using a basic teach-in procedure.
  Scenario 2: Limited modifications brought to a robot program in the context of an advanced robotic automation application, which already provides means for autonomous manipulation of work pieces and error recovery in case of collisions, whereby the envisioned error recovery process partly requires human intervention.

Whereas in the first scenario, the focus will be on the ease and speed of programming a cobot, the second scenario illustrates how already programmed and possibly certified applications can be modified during productive use by shop floor employees without interrupting production for a long period.

3.1. Scenario 1: Faster Cobot Programming

Fig. 3 shows the setup of the simple pick-and-place application. The evaluation was conducted with the help of 4 human participants, who did not know how to program the Panda robot before. The participants were first introduced in how to use Panda's native HPI and the asked to program the pick-and-place task depicted in Fig. 3 using the robot's HPI, which was running on a laptop. The participants used a mouse and the laptop's keyboard to program the task, which consisted in picking and placing 4 screws. Then, the same participants were introduced to the programming by voice extension to the robot's HPI and asked to program the same task without using the mouse and keyboard. This video demonstrates how this task is performed using voice programming: [10].

Fig. 4 shows a breakdown of the operation durations with respect to the three groups of supported voice commands and their mouse and keyboard equivalents. In addition, the time spent manipulating the robot and waiting for the robot's HPI to react were also considered in the evaluation. These results show that, for the tested scenario, programming by voice combined with the haptic manipulation of the robot was 46% faster than using the robot's native HPI, whereby most of the time was saved with not having to set app parameters and clicking through the wizards when using the speech mode.

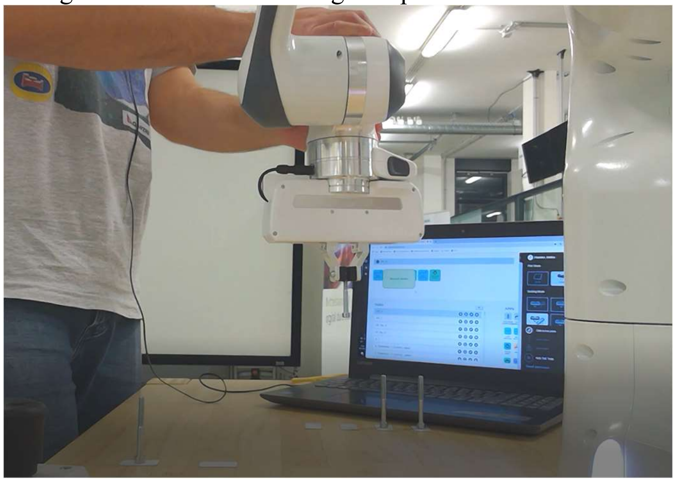

Fig. 4. Breakdown of operation durations in scenario 1.

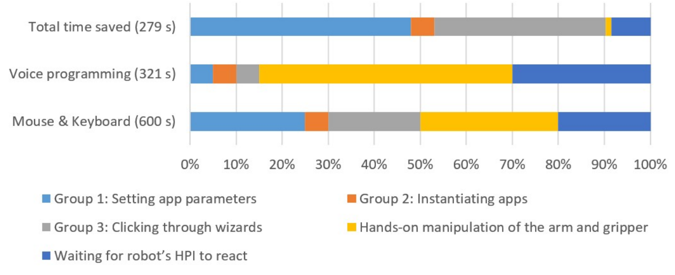

Fig. 4. Breakdown of operation durations in scenario 1.

3.2. Scenario 2: Error Recovery Through Human Intervention

The second application scenario consists in a chess playing program, which is able to recover from an error by requesting human intervention. If the robot fails to grasp a piece due to a collision, the program stops and waits for human intervention. In preparation of this intervention, the program automatically creates a new task, which opens the gripper wider in order for the robot to grasp the piece correctly. The human is only required to set the width of the gripper so that it will be able to grasp the piece without disturbing any other pieces on the board. The following video demonstrates this scenario: [11].

To accomplish this task, the user first switches the mode from operational to teach-in by pulling out the safety stop button. Then the user repositions the gripper fingers and issues the "Okay" command, which autocompletes the remaining wizard steps and resumes the program. Hence, the only interactions between the human and the robot are voice-based and haptic.

This enables a safety-certified application to allow minor program changes, which do not affect the results of the risk analysis of the application.

4. Conclusion

This paper introduced a new method and tool for programming cobots by voice. This provides an additional mode of programming cobots, which currently offer multi-modeal programming environments based on haptic, mouse, and keyboard interactions. The main benefit of this approach is an increase in the productivity of cobot programmers in scenarios which require a high number of teach-in operations. In addition, we explored a scenario in which shop floor employees can perform small modifications to existing programs, without requiring recertification. Another benefit of the approach is the possibility of communicating cobot programs by demonstration through videos--a feature that may prove useful in training and education contexts.

4.1. Limitations and Future Work

The current implementation of the programming by voice extension to Franka Emika Panda's HPI has the following limitations:

- Only teach-in commands are supported. In a future version, more voice commands will be supported, which will allow users to program, for example, coordinate based motions.
- Currently, only the web-based HPI of the Panda robot is supported. However, any other web-based HPI can be easily endowed with voice programming support using the same technique. Other kinds of user interfaces can also be extended by voice programming, for example, using GUI automation tools.
- The assembly precision achieved through voice based programming is currently below that offered by traditional means of robot programming. The current approach was specifically aimed at speeding-up teach-in tasks, which is one popular means of programming cobots.
- Additional limitations may exist depending on the application at hand.

Acknowledgements

This research was co-funded by the Austrian Research Promotion Agency (FFG) as part of the HCCPPAS grant project.

References

[1] Ionescu, T. B., & Schlund, S. (2019). A Participatory Programming Model for Democratizing Cobot Technology in Public and Industrial Fablabs. Procedia CIRP, 81, 93-98.

[2] Makris, S., Tsarouchi, P., Surdilovic, D., & Krüger, J. (2014). Intuitive dual arm robot programming for assembly operations. CIRP Annals, 63(1), 13-16.

[3] Weintrop, D., Shepherd, D. C., Francis, P., & Franklin, D. (2017). Blockly goes to work: Block-based programming for industrial robots. In 2017 IEEE Blocks and Beyond Workshop (B&B) (pp. 29-36). IEEE.

[4] Heimann, O., & Krüger, J. (2018). Affordance Based Approach to Automatic Program Generation for Industrial Robots in Manufacturing. Procedia CIRP, 76, 133-137.

[5] Beschi, S., Fogli, D., & Tampalini, F. (2019, July). CAPIRCI: a multi-modal system for collaborative robot programming. In International Symposium on End User Development (pp. 51-66). Springer, Cham.

[6] Bauer, W., Schlund, S., & Vocke, C. (2017, July). Working life within a hybrid world–how digital transformation and agile structures affect human functions and increase quality of work and business performance. In International Conference on Applied Human Factors and Ergonomics (pp. 3-10). Springer, Cham.

[7] Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In Proceedings fifth ieee international enterprise distributed object computing conference (pp. 118-127). IEEE.

[8] Begel, A., & Graham, S. L. (2006, September). An assessment of a speech-based programming environment. In Visual LAnguages and Human-Centric Computing (VL/HCC'06) (pp. 116-120). IEEE.

[9] Arnold, S. C., Mark, L., & Goldthwaite, J. (2000, November). Programming by voice, VocalProgramming. In Proceedings of the fourth international ACM conference on Assistive technologies (pp. 149-155). ACM.

[10] Scenario 1 video: https://drive.google.com/open?id=1sgM5Bop7mjEFgtq8C1670nqNhROTJ8tc

[11] Scenario 2 video: https://drive.google.com/open?id=1sgM5Bop7mjEFgtq8C1670nqNhROTJ8tc

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

8th CIRP Conference of Assembly Technology and Systems

Programming Cobots by Voice: A Human-Centered, Web-Based Approach

Tudor B. Ionescu [a]*, Sebastian Schlund [a]

[a]*Human-Machine Interaction Group, Institute of Management Science, Technische Universität Wien; Theresianumgasse 27, 1040-Vienna, Austria.*

* Corresponding author. Tel.: +43 (1) 58801-33057; fax: +43 (1) 58801-33098. E-mail address: tudor.ionescu@tuwien.ac.at

Abstract

We present a new voice-based programming approach and software framework for collaborative robots (cobots) based on the Web Speech API, which is now supported by most modern browsers. The framework targets human programmable interfaces (HPIs), which can be used by people with little or no programming experience. The framework follows a meta-programming approach by enabling users program cobots by voice in addition to using a mouse, tablet or keyboard. Upon a voice instruction (such as move, pick, release, etc.), the framework automates the manual tasks required to manipulate the vendor-provided HPI. The main advantages of this approach are simplified, guided programming, which only requires the knowledge of 5–10 voice instructions; increased programming speed by up to 46% compared to the manual approach; and the possibility of sharing programs as videos. The open-source framework was evaluated using two application scenarios.

Keywords: Type your keywords here, separated by semicolons ;

1. Introduction

Assembly automation has increasingly developed over the last decades also thanks to the introduction of collaborative robotic arms (short, cobots), which can work in close proximity to humans without requiring a safety fence. Cobots are currently used to automate repetitive manual tasks, such as loading/unloading of machines, pick & place operations, screwing tasks, or quality inspection. Cobots can help to accomplish some of these tasks more cost effectively than humans in productive industrial contexts. Nevertheless, the field of hybrid automation—i.e., where humans and machines work together in direct interaction has not been realized to the extent projected by the industrial automation community and cobot vendors.

Currently, a number of factors sum up to prevent the wide scale adoption of cobots in Europe. First, in most European countries, each application requires its own safety certification, which prevents users to change the application layout and program without conducting a new risk assessment. Second, current robot programming environments are usually considered counter-intuitive because they build upon older expert programming models, such as function blocks, textual programming, and combinations of visual and textual programming modes. At the same time, newer robot programming environments, which draw upon design patterns used for smart phone user interfaces and other commonly used day-to-day technologies, provide opportunities for non-experts to engage with programming cobots more easily [1]. Yet, while the barriers are lowered, these environments often lack the versatility of textual programming and are subjected to app-oriented business models, which foresee the acquisition of additional robot apps (or skills) at prohibitive costs. And third, while robot vendors strive for targeting new markets and users, in the industrial domain, robot programming is still regarded as being a task for experts. This leads to a gap concerning a learner’s transition from multi-model, intuitive robot programming towards the more versatile, textual programming environments. Such a transition would, for example, allow assembly workers to program robots using increasingly complex programming models.

In response to this situation, we envision a scenario in which robot programming is certified as an assembly application instead. This would allow for workers to design applications in collaboration with assembly planners without the need for recertification. In this context, the collaboration between humans and robots occurs at the cognitive level, with workers and other non-experts designing and programming applications. Then, depending on lot size, certification costs, and other factors, organizations may choose to operate cobots behind a fence or not. In any case, all employees would benefit from directly programming cobots on the long term, as industrial robot safety research is expected to lower the barriers required to certify industrial cobot applications.

Drawing on the assumption that the notion of intuitive user interfaces is contingent on the user's prior knowledge and experience with technologies having similar user interfaces as well as with analogous day-to-day activities (i.e., conversations, text chatting, web searching, interactions with smart phones and tablets, etc.) we hypothesize that intuitive robot programming is not a holy grail to be attained by one human-programmable interface (HPI); but a matter of providing diverse supports for that which is already known and experienced by the projected users. These supports may include app-oriented drag-and-drop features, voice-based programming, and other multimodal interactions.

In this paper, we dive into voice-based programming as one mode of supporting a more intuitive interaction with cobots. This feeds into the scenario in which cobot programming is safety-certified as an industrial application and workers can accomplish a great number of programming tasks on the shop floor. We begin by discussing cobot programming by voice in the broader context of speech-based programming. Then, we introduce a new concept and lightweight tool for voice-based cobot programming, which leverages existing technologies; and then evaluate it on the basis of two usage scenarios.

1.1. (Robot) Programming by Voice

Programming by voice or speech is a compelling idea, which received some attention by the software engineering and robotics communities in the past 25 years. While during the 2000s the main problem appeared to be the lack of reliable speech recognition techniques, owing to the recent advances is machine learning, today researchers are confronted with other problems, such as how to structure voice commands in order to produce code while minimizing the necessary corrections [8,9]. In this sense, to reduce the speech recognition error, some researchers use code words instead of 1-to-1 mappings between what is being said/recognized and what is being coded. In the industrial robotics domain, programming by voice (i.e., producing program code rather than controlling or guiding a robot by voice) has received relatively little attention compared to other programming techniques, such as block-based [3], automated [4] programming, or multimodal teach-in [5]. More work appears to have been invested in controlling or guiding a robot using voice commands [2], whereby these commands are used to immediately call some pre-programmed function of the robot (e.g., different movements or invocation of more complex robotic skills). In the case of collaborative industrial robots (cobots), "conversations" between humans and robots on the shop floor are often very limited and can sometimes even be annoying, as the buzz of the factory is interfering with the intentions of the user. The advantages of operatively controlling cobots by voice are also limited by the requirement to certify cobot applications individually, which leaves little room for creativity during operation. In this context, simple buttons are often more effective in triggering or acknowledging cobot actions in productive applications.

Against this background, this paper presents a novel lightweight cobot programming by voice framework based on the new Web Speech API (WSAPI)—a W3C specification published and maintained by the Speech API Community Group supported by modern web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge. The framework explicitly targets the assembly application development phase, which usually requires many rounds of repetitive trial and error until robots reach the necessary precision, especially in small part assembly, such as PCB assembly. In this context, programming and testing cobot applications by voice can be more effective in terms of speed and difficulty. This would also empower shop floor employees to participate in the application development activities, as a growing body of work has called for (e.g., [1,6]).

The framework we developed uses the WSAPI to wrap the programming structures used by a cobot produced by the Franka Emika company, which already offers a simplified web-based human programmable interface (HMI). In this environment, non-experts can program the robot by dragging, dropping, and configuring predefined apps in a sequence or according to simple patterns, such as repetitions of the same action or more complex composite apps, which can be acquired in an app store. This app-oriented way of programming cobots appears to be more intuitive than other industrial cobot HPIs because it draws on known patterns of interactions (e.g., with smart phones and tablets). This provides non-programmers with the opportunity of a fast and easy initiation into cobot programming. Nevertheless, the same programming technique becomes rather annoying and slow once a person is initiated into the craft of cobot programming. In this context, programming by voice can help to bridge the potential interest gap, which may arise when a person finds something to be too easy and limited. Also, and perhaps more importantly from an economic point of view, programming cobots by voice can boost the productivity of both beginner and expert programmers by allowing them to use their hands for guiding the robotic arm rather than inputting text or dragging apps and functions. The framework introduced in this paper thus aims to augment the multimodal programming capabilities of existing cobots rather than providing a definitive replacement for any of them. Also, while the framework currently only support the Franka Emika Panda robot, we are already working on a version for the Universal Robot series, which will become available in the course of 2020.

2. Cobot Programming by Voice: A Web Framework

This section provides technical details about the design and implementation of the web-based programming by voice framework discussed in this paper. We begin by introducing the Panda robot's native HPI and the WSAPI before describing the software components needed to extend the capabilities of this interface.

2.1. Programming the Panda Cobot: Features & Limitations

The Franka Emika Panda collaborative, industrial-grade robotic arm is one of the first of its kind to provide a web-based HPI. The design of this HPI closely follows the model-view-controller (MVC) pattern [7], with a REST and Websockets backend and a dynamic HTML5 & JavaScript web interface. The REST service components of Panda's HPI control and maintain the robot's state by processing sensor signals and actuating robot moves. The backend application, which implements the model in the MVC pattern, wraps the robot's native C++ API, called libfranka. This model maintains the current state of the robot (i.e., joint positions, applied forces, sensor values, actuation history, etc.) in memory. The web application, shown in Fig. 1, is updated via web-sockets to reflect the current state of the model. In this web environment, users can create new robot applications by dragging and dropping so-called robot apps from an app collection to a workflow pane to form a sequence of basic actions, whereby each app in the sequence represents a configurable instance of a movement or behavior of the robot (e.g., taught-in or relative moves, gripper actions, waiting for user input, etc.). In addition, there exist so-called pattern apps, which implement predefined applications patterns (e.g., picking and/or placing parts in a tray, or more advanced program structures). Named app sequences, which are called tasks, are persisted in the model, whereby all user interactions generate calls to the REST services.

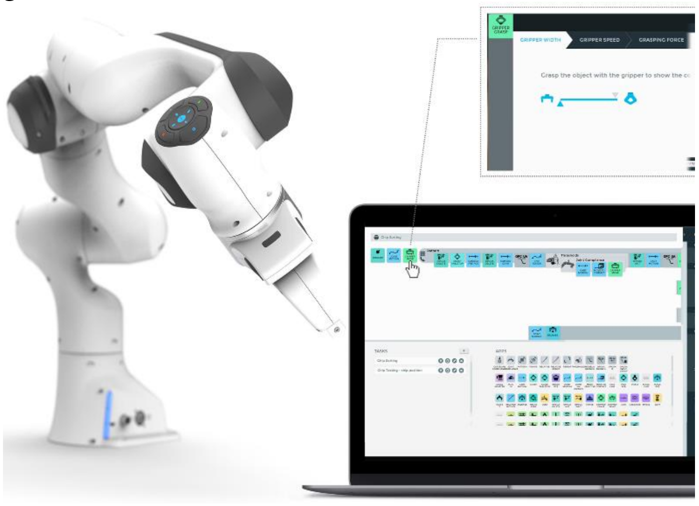

Fig. 1. The Franka Emika Panda robot and HPI.

This app-oriented HPI is coupled with a series of buttons available on the robot's head, as shown in Fig. 1. Two of these buttons are used to move the arm in free-drive mode. The other ones are used within the wizard-like configuration of each app, which is activated by clicking on the respective app, as shown in Fig. 1. This way the user only needs to interact with the laptop or tablet when dragging and dropping a new app into the workflow. For most of the available apps, all other parameters can be configured using the buttons from the robot's head. In practice, however, users prefer to set certain parameters and click through the wizards using the mouse or touch screen of the laptop of tablet because the buttons are relatively hard to press and thus induce physical strains into the wrist when used repeatedly for a long time. Also, configuring the apps using the buttons is usually slower than using the mouse and keyboard. Hence, while Panda's HPI arguably represented a leap in intuitive robot programming, it has some obvious limitations when it comes to ergonomics and speed.

In our experience with using this robot in research and teaching activities, these limitation not only reduce the productivity of users, who need to switch back and forth between the robot's head and the computer, but may also pose a safety hazard when more than one individual works with the robot. Students, for example, tend to divide tasks so that one person will more the robot, while another one configures the apps. At times, however, both persons may stare at the computer screen, whereby one of them (the one who guided the robot's head) may be in close proximity to the robot, which—on occasion—may perform unexpected moves (e.g., in case of a software or hardware error) and thus potentially harm users. As we argued elsewhere, when working in teams of two, only one person should program the robot, while the other one should watch that all safety-related rules are being followed [1]. These roles should be switched periodically.

2.2. Leveraging the Web Speech API to Enable Cobot Programming by Voice

The WSAPI is a collection of client-sided JavaScript functions embedded in modern web browsers (notably Chrome and Firefox), which allows web developers to leverage the newest machine-learning based speech recognition technologies. Common use cases include voice web search, speech command interface, speech translation, dialog systems, multimodal interaction and search, etc. The WSAPI can access the computer's microphone (or any other microphone connected to it) via the browser. To ensure privacy, the user is asked for permission before listening is enabled. As opposed to older speech recognition frameworks, the WSAPI supports a wide variety of languages and has a very high recognition success rate. This makes corrections only necessary in the presence of external acoustic interferences, provided that the user is a fluent speaker in his/her language of choice.

Once enabled, the WSAPI listens to what is being said and converts spoken sequences into words placed, for example, in an array. To implement cobot programming by voice, we developed a script that can be injected into Panda's web-based HPI using the "bookmarklet" technique. A bookmarklet is a bookmark containing JS commands stored in the browser's "favorites" bar, which adds "one-click" functionalities to the currently displayed web page. This technique has been effectively used to embed or extend programming environments in existing webpages (e.g., [20]). The current approach uses the same mechanism to inject JavaScript code into Panda's HPI, which automates the instantiation, loading, configuration, and execution of apps and tasks in the robot's native HPI according to the user's voice commands.

The bookmarklet technique we used to extend Panda's HPI by voice programming falls into the broader category of graphical user interface (GUI) automation, which is primarily used in software testing. Yet, with the increasing diversity and complexity of software used in companies and privately, more GUI automation use cases emerged in the past few years. For example, the award winning UiPath toolset supports the development of so-called software robots that automate repetitive, annoying user interactions with software systems such as SAP or Microsoft Excel. In this spirit, the proposed framework aims to bring this new philosophy of interfacing with existing software GUIs to the world of collaborative robotics programming to the end of boosting productivity.

In the following we provide some technical details about the implementation of the framework before turning to the an empirical evaluation and discussion of the approach.

2.3. Framework Design and Features

Fig. 2 illustrates the architecture of the “Programming by Voice” extension to the Panda robot’s HPI. Building on the robot’s HPI, which implements the well-known Model-View-Controller (MVC) design pattern, the user injects a so-called “Programming by Voice” (PBV) controller into the current browser page (i.e., the robot’s HPI shown in Fig. 1). Once injected, the PBV controller activates and maintains an open connection to the Web Speech API embedded in the browser, which enables it to listen for voice commands coming from the user. Upon recognizing one of the commands listed in Table 1, the PBV controller may use the visual controls available to users by automating the use actions needed to accomplish certain tasks (e.g., to drag, drop, or configure different robot behavior apps). The PBV controller can also directly manipulate the server-side components of the system by sending REST service calls. One the server side, the application model (which contains the robot’s state, programmed tasks, and app implementations) is maintained and persisted in memory. The robot’s HPI is thus augmented by a new programming mode, which enables users to create tasks while keeping their hands free (e.g., for teaching positions by guiding the robot and configuring the gripper using both hands). This eliminates the need for switching back and forth between the robot and the laptop or tabled running its HPI.

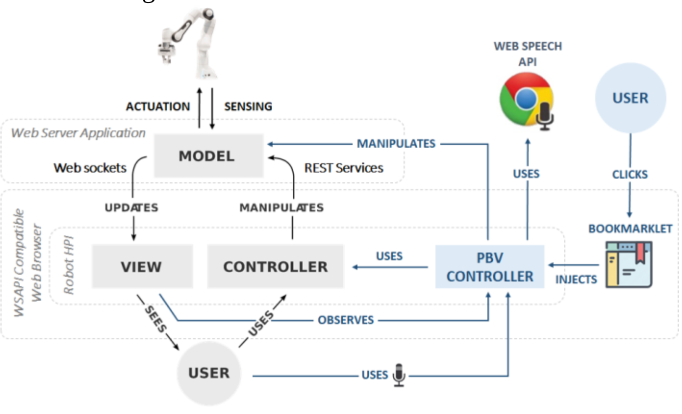

Fig. 2. Architecture of the “Programming by Voice” extension.

The following commands are supported in the current implementation, which can be divided into three distinct groups, depending on the context of use:

Group	Command	Functionality
1	*Speed/Force/ Load/Speed-up <#>*	Sets the velocity / grasping force / gripper load / acceleration of the arm for all subsequent apps to the specified integer value <#>.
2	*Grasp*	Initializes the “Gripper Grasp” app which allows users to manually set the opening of the gripper so as to grasp a certain object. After triggering this app, the users are able to use the buttons on the robot’s head or to manually set the gripper fingers to the desired position.
	Hand	Initializes the “Gripper Move” app which allows users to set the opening of the gripper using the buttons on the robot’s head or by manually moving the gripper fingers.
	Motion	Initializes the “Cart Motion” app, which allows users to teach in a movement of the arm using several waypoints. Waypoints are set by driving the robot’s arm to the desired pose and then pushing the OK button on its head.
3	*Okay*	After having set all the parameters of an app by using the buttons on the robot’s head and/or manually moving the robot’s arm and fingers, this command will auto-complete all the information requested in the subsequent dialogues of the app-specific wizard (e.g., velocity, force, acceleration, load).
	Start	This command starts the execution of the task being programmed.
	Stop	This command stops the execution of a task.

The first group of commands can be issued at any time while working with the robot’s native HPI. These commands save time and clicks by allowing users to set reusable parameters before configuring specific apps. While configuring an app, which requires certain parameters, issuing the “Okay” command will automatically set the values of the app-specific parameters to those of the matching global parameters. If users forget to set these global parameters to the desired value, they can still be changed at a later time using the app’s configuration wizard.

The second group of commands instantiate apps for actuating the robot’s arm and tool (e.g., gripper). In the robot’s native HPI, these off-the-shelf apps need to be dragged from the app library and dropped in the workflow pane of the currently active task. The task of instantiating apps, which can take 5-10 seconds per app, can be automated using voice commands. This saves time because, while the robot’s HPI performs the initialization of a task, the user can already drive the robot into the desired pose for the first waypoint of the motion or set the width of the gripper fingers. After all waypoints are configured using the buttons on the robot’s head and/or the gripper width is manually set by moving its fingers, the user can issue the “Okay” command to autocomplete the remaining app parameters, such as the velocity and acceleration of a movement and the force and load for pick and place operations.

The third groups of commands is used to trigger actions, which would require one or several mouse clicks or tapping the touch screen of the device running the robot's HPI (e.g., laptop, tabled, etc.). The "Okay" command triggers a context-dependent series of actions so as to autocomplete the parameters of the app being configured and to automatically click-through all remaining dialogues of the respective app configuration wizard. The "Start" and "Stop" commands will simply start and stop the currently active task. By using voice commands the user can focus on interacting with the robot through haptic means. To switch from the task programming into the task testing and running mode, the user needs to pull up the robot's safety stop button, which is pushed down during teaching and programming. Hence, using the start/stop commands, users can save time, for example, by allowing users to focus on the robot's actions during testing while keeping the safety stop button in their hands. This also contributes to a safer interaction with the robot during testing, since the user's attention is not distributed between the robot arm and HPI screen.

3. Evaluation of the Approach

In this section we present an evaluation of the programming by voice concept and extension on the basis of two application scenarios, which assume that shop floor employees (e.g., assembly workers and planners) engage in programming the robot in the following situations:

- **Scenario 1:** Fast programming of pick and place operations for small lot sizes using a basic teach-in procedure.
  Scenario 2: Limited modifications brought to a robot program in the context of an advanced robotic automation application, which already provides means for autonomous manipulation of work pieces and error recovery in case of collisions, whereby the envisioned error recovery process partly requires human intervention.

Whereas in the first scenario, the focus will be on the ease and speed of programming a cobot, the second scenario illustrates how already programmed and possibly certified applications can be modified during productive use by shop floor employees without interrupting production for a long period.

3.1. Scenario 1: Faster Cobot Programming

Fig. 3 shows the setup of the simple pick-and-place application. The evaluation was conducted with the help of 4 human participants, who did not know how to program the Panda robot before. The participants were first introduced in how to use Panda's native HPI and the asked to program the pick-and-place task depicted in Fig. 3 using the robot's HPI, which was running on a laptop. The participants used a mouse and the laptop's keyboard to program the task, which consisted in picking and placing 4 screws. Then, the same participants were introduced to the programming by voice extension to the robot's HPI and asked to program the same task without using the mouse and keyboard. This video demonstrates how this task is performed using voice programming: [10].

Fig. 4 shows a breakdown of the operation durations with respect to the three groups of supported voice commands and their mouse and keyboard equivalents. In addition, the time spent manipulating the robot and waiting for the robot's HPI to react were also considered in the evaluation. These results show that, for the tested scenario, programming by voice combined with the haptic manipulation of the robot was 46% faster than using the robot's native HPI, whereby most of the time was saved with not having to set app parameters and clicking through the wizards when using the speech mode.

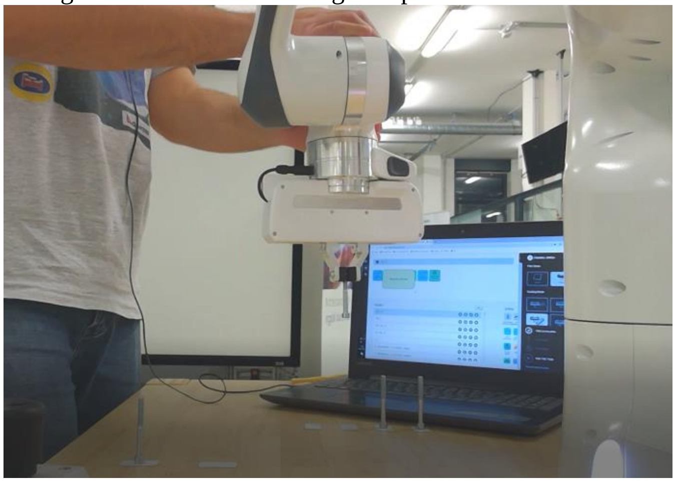

Fig. 4. Breakdown of operation durations in scenario 1.

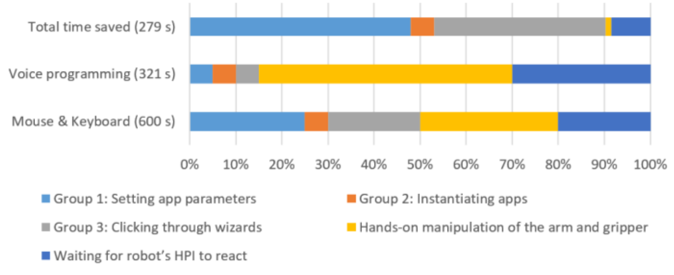

Fig. 4. Breakdown of operation durations in scenario 1.

3.2. Scenario 2: Error Recovery Through Human Intervention

The second application scenario consists in a chess playing program, which is able to recover from an error by requesting human intervention. If the robot fails to grasp a piece due to a collision, the program stops and waits for human intervention. In preparation of this intervention, the program automatically creates a new task, which opens the gripper wider in order for the robot to grasp the piece correctly. The human is only required to set the width of the gripper so that it will be able to grasp the piece without disturbing any other pieces on the board. The following video demonstrates this scenario: [11].

To accomplish this task, the user first switches the mode from operational to teach-in by pulling out the safety stop button. Then the user repositions the gripper fingers and issues the "Okay" command, which autocompletes the remaining wizard steps and resumes the program. Hence, the only interactions between the human and the robot are voice-based and haptic.

This enables a safety-certified application to allow minor program changes, which do not affect the results of the risk analysis of the application.

4. Conclusion

This paper introduced a new method and tool for programming cobots by voice. This provides an additional mode of programming cobots, which currently offer multi-modeal programming environments based on haptic, mouse, and keyboard interactions. The main benefit of this approach is an increase in the productivity of cobot programmers in scenarios which require a high number of teach-in operations. In addition, we explored a scenario in which shop floor employees can perform small modifications to existing programs, without requiring recertification. Another benefit of the approach is the possibility of communicating cobot programs by demonstration through videos--a feature that may prove useful in training and education contexts.

4.1. Limitations and Future Work

The current implementation of the programming by voice extension to Franka Emika Panda's HPI has the following limitations:

- Only teach-in commands are supported. In a future version, more voice commands will be supported, which will allow users to program, for example, coordinate based motions.
- Currently, only the web-based HPI of the Panda robot is supported. However, any other web-based HPI can be easily endowed with voice programming support using the same technique. Other kinds of user interfaces can also be extended by voice programming, for example, using GUI automation tools.
- The assembly precision achieved through voice based programming is currently below that offered by traditional means of robot programming. The current approach was specifically aimed at speeding-up teach-in tasks, which is one popular means of programming cobots.
- Additional limitations may exist depending on the application at hand.

Acknowledgements

This research was co-funded by the Austrian Research Promotion Agency (FFG) as part of the HCCPPAS grant project.

References

[1] Ionescu, T. B., & Schlund, S. (2019). A Participatory Programming Model for Democratizing Cobot Technology in Public and Industrial Fablabs. Procedia CIRP, 81, 93-98.

[2] Makris, S., Tsarouchi, P., Surdilovic, D., & Krüger, J. (2014). Intuitive dual arm robot programming for assembly operations. CIRP Annals, 63(1), 13-16.

[3] Weintrop, D., Shepherd, D. C., Francis, P., & Franklin, D. (2017). Blockly goes to work: Block-based programming for industrial robots. In 2017 IEEE Blocks and Beyond Workshop (B&B) (pp. 29-36). IEEE.

[4] Heimann, O., & Krüger, J. (2018). Affordance Based Approach to Automatic Program Generation for Industrial Robots in Manufacturing. Procedia CIRP, 76, 133-137.

[5] Beschi, S., Fogli, D., & Tampalini, F. (2019, July). CAPIRCI: a multimodal system for collaborative robot programming. In International Symposium on End User Development (pp. 51-66). Springer, Cham.

[6] Bauer, W., Schlund, S., & Vocke, C. (2017, July). Working life within a hybrid world–how digital transformation and agile structures affect human functions and increase quality of work and business performance. In International Conference on Applied Human Factors and Ergonomics (pp. 3-10). Springer, Cham.

[7] Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In Proceedings fifth ieee international enterprise distributed object computing conference (pp. 118-127). IEEE.

[8] Begel, A., & Graham, S. L. (2006, September). An assessment of a speech-based programming environment. In Visual LAnguages and Human-Centric Computing (VL/HCC'06) (pp. 116-120). IEEE.

[9] Arnold, S. C., Mark, L., & Goldthwaite, J. (2000, November). Programming by voice, VocalProgramming. In Proceedings of the fourth international ACM conference on Assistive technologies (pp. 149-155). ACM.

[10] Scenario 1 video: https://drive.google.com/open?id=1sgM5Bop7mjEFgtq8C1670nqNhROTJ8tc

[11] Scenario 2 video: https://drive.google.com/open?id=1sgM5Bop7mjEFgtq8C1670nqNhROTJ8tc