Implementing Human-Robot Interaction to Mimic Human Writing

Tomasz KORDZIUKIEWICZ [a], Anastasiya KREIDZICH [b] , Moises DIAZ [c] and Kanstantsin MIATLIUK [a]
[a] *Bialystok University of Technology, Robotics and Mechatronics Department, WM*
Wiejska 45C, 15-351, Bialystok, POLAND
[b] *State University of Grodno*
22 Ozheshko str., 230023, Grodno, BELARUS
[c] *University Las Palmas Gran Canaria*
Campus Universitario de Tafira, 35017, Las Palmas, SPAIN
tomekkordziukiewicz@gmail.com, forsajozer@mail.ru, moises.diaz@ulpgc.es, k.miatliuk@pb.edu.pl

Abstract. The paper presents an approach to implementing human-robot interaction via computer to perform writing process by UR3 robot mimicking human writing. To this aim, online human signing standards were generated first. Next, the human-robot communication technologies were tested and the novel author's method of interaction with robot via computer was proposed to perform robot writing. After that, robot writing task was performed using human signing standards mimicking human writing and robot signatures were acquired as a result. Finally, conclusive remarks were presented.

1. Introduction

The study presented in the paper aimed at the development and implementation of a novel method of human-robot interaction via computer to perform robot writing. For this aim the laboratory stand (Figure 1) was employed in Robotic Systems Lab., Bialystok University of Technology, Poland. To fulfil the robot writing using this stand it is necessary to perform several actions at the following phases of the robot writing process.

At the first phase, the writing standard of human signature was acquired. The acquisition process was performed similarly as it was described in (Miatliuk & al., 2019).To obtain the writing standard human operator wrote the predefined word using Wacom Intuos Pro tablet (Figure 1) connected with control PC. Computer program wacomGUI2 (WacomGUI2, 2021) installed on PC allowed processing and collecting of the obtained human writing data in correspondent *csv* files.

At the second phase, communication technologies were analysed and the author's method was proposed to implement human-robot interaction via computer to perform UR3 writing. Three potential solutions were analysed, i.e. RoboDK program system (RoboDK, 2022), ROS system with industrial package (ROS, 2021), and Python program (Python, 2022) to communicate with robot. Next, Python program was chosen to fulfil the computer-robot interaction. Novel scripts for data converting and communication were developed and merged.

At the third phase, robot writing task was performed by implementing human-robot interaction method proposed using human signing standards generated. The human writing data acquired in the first phase was converted with the help of Python program module developed and used for the realisation of robot writing at the third phase. Converted data were presented in the form of table of points coordinates on XY plane and two positions meanings, i.e. 0 or 1, along Z axis. Communication class of Python was used to interact with UR3 robot (UR3 user manual, 2022) and generate URScript commands (URScript, 2021) based on the converted data. The robot writing processes were successfully performed and the robot signatures were acquired as a result.

The human and the robot signatures can be further analyzed and compared. The degree of closeness of the human and robot writing can be defined using the methods described in (Miatliuk & al., 2019; Wolniakowski & al., 2021). In this work, the human and robot writing trajectories were compared visually by the authors.

a) 

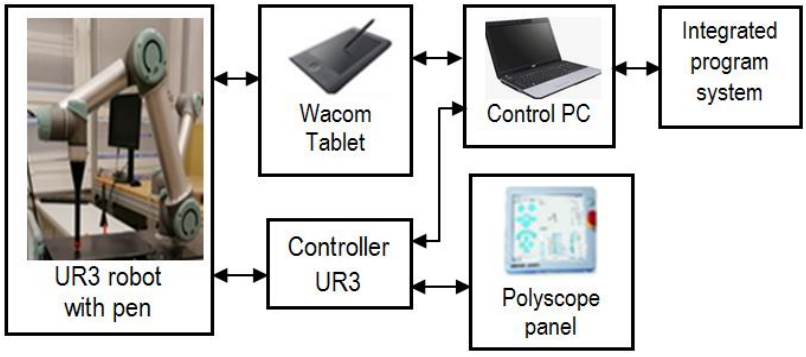
b) 

Figure 1. a) Laboratory stand, b) Subsystems interactions of the laboratory stand in a robot writing process.

2. Acquisition of human writing standard

The research started by human writing standard acquisition**.** To obtain the standard the Wacom Intuos Pro L tablet (Figure 1) which is connected with control PC is used. Computer program wacomGUI2 installed on control PC allows processing and writing of the obtained human writing data in correspondent *csv* files.

The data in *csv* file is presented in a seven columns table form (Figure 2). Columns *1* and *2* present *X* and *Y* coordinates of human signatures given in coordinate system of tablet. Change of *X* or *Y* means the tablet pen displacement while writing. Coordinates values are written in a format dependent on DPI parameter of the tablet. To present coordinates values in meters the scaling of these values is further performed. The time *t* of the writing process is presented in column *3* of *csv* file, and the pen pressure information *Z* is given in column *4*. Information presented in columns *5-7*, such as a *tilt* of the tablet pen, is not used in the robot writing process under consideration.

Data obtained from Wacom tablet using wacomGUI2 program and presented in the *csv* data file described above is further converted by Python control program developed in the work to calculate end-effector (robot pen) positions of UR3 to perform a robot writing process according to the human writing standard acquired. In this way, data obtained from tablet is further used in robot control processes while writing.

X	Y	t	Z	pen tilt		
7174	31476	210164820	1	1250	430	118
7174	31476	210164825	1	1260	440	146
7174	31476	210164830	1	1260	440	153
7174	31476	210164835	1	1260	440	163
7174	31476	210164840	1	1260	440	176
7174	31476	210164845	1	1260	440	189
7174	31476	210164850	1	1260	440	189
7174	31476	210164855	1	1260	440	175

Figure 2. Data fragment of *csv* file.

3. Implementing human robot interaction via computer

Implementing human-robot interaction by connecting robot and human-operator via computer was the main task of the presented research. Three methods were tested to perform the task. First and potentially easiest way was the using of the RoboDK program system (RoboDK, 2022). To establish computer-robot connection RoboDK program needs robot IP address only. Robot programming can be done by RoboDK system using URScript or Python script. But this method wasn't selected due to the problems of RoboDK and UR3 communication. Implementation of ROS system with ROSindustrial package (ROS, 2021) is the second interaction method which was tested. The package provides various robot models ready to use. Almost ready solution shared on GITLab (Wolniakowski. 2014) could provide the best way to perform the task. Unfortunately, the compatibility and communication problems of the hardware and software subsystems available in the experimental study did not allow implementation of this method. Therefore, development of a novel Python program module was conducted in the work to implement human-computer-robot interactions to perform robot writing.

Development of a novel author's Python program module using the communication node of ROS package (ROS operating system, 2021) was the main task of this phase of the work. To implement human-robot interactions it was essential to understand how communication between UR3 and computer could be realised. The data frame presented in Figure 3 was developed by authors (Kordziukiewicz, 2022) based on ROS package and UR program modules to implement computer-robot communication. Sending message to robot from computer wasn't a problem. To receive information from robot via TCP/IP protocol it was necessary to decode the developed data frame. Each segment of the data frame contained the robot information, such as joint position, joint forces, etc. The data frame starts with five bytes segment which contains the information about the whole length of the data frame (4 bytes) and the robot type information (1 byte). Next four bytes segment informs about sub-package length. The following one-byte segment contains the information about the sub-package type, e.g. robot joint data. The next segment (8 to 48 bytes length) contains the concrete information presented by the sub-package, e.g. the position of every robot's joint. The length of sub-package depends on the information type it contains. The whole frame contains every sub-packages type and each of them has the same pattern of the information coding.

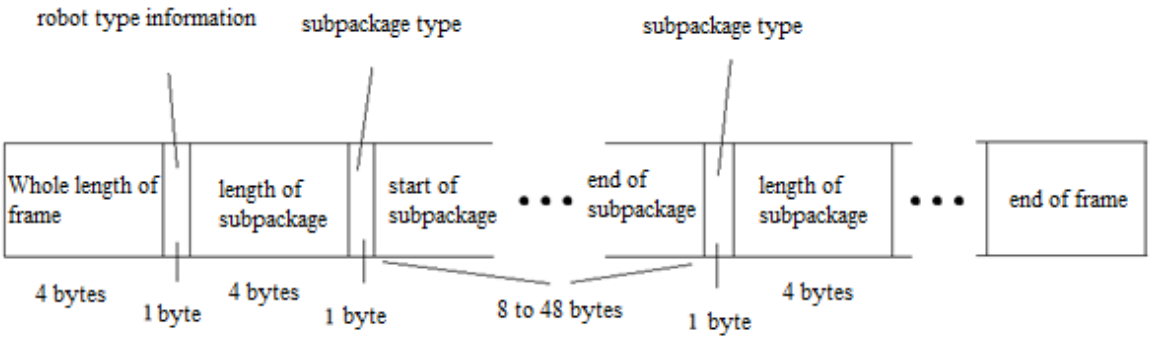


Figure 3. Robot-computer communication data frame.

To decode the data frame the author's *handleMessage()* function was created and used to sort the data by type of information. The sorted data was transferred to another function which decoded the sub-package information and presented the information in the required form. The example of the decoding function script is shown in Figure 4. Connection between computer (laptop) and robot was possible by using Python *socket* and *threading* libraries. By using *sockets* functions it was possible to access low-level communication between robot driver and computer. Developed communication script demanded robot IP address only to gain the robot-computer interaction.

```
def handleJointData(self, data):
    for i in range(0, 6):
        idx = 41 * i
        self.joint_positions[i] = struct.unpack('>d', data[idx + 5:idx + 13])[0]
        self.joint_speeds[i] = struct.unpack('>d', data[idx + 21:idx + 29])[0]
        self.joint_currents[i] = struct.unpack('>f', data[idx + 29:idx + 33])[0]
        self.joint_voltages[i] = struct.unpack('>f', data[idx + 33:idx + 37])[0]
        self.joint_modes[i] = data[idx + 45]
```

Figure 4. Example of decoding function of sorted robot joints data, i.e.position, speed, currents, etc .

4. Performing robot writing

Last phase of the research was the performing of writing process by robot UR3. In this process, the data of the main Python script developed was converted from columns into tables format which is useful in generating robot movement. The tablet pen and XY coordinates data were transferred by the main script to the communication program module to generate URScript. That module used data from *svc* file to generate XYZ movement of robot end-effector, i.e. robot pen. Robot motion parameters such as velocity, acceleration, etc. were chosen experimentally and can be changed in the created Python scripts. The example of *movej* command sending by the developed communication program module to UR3 is presented in Figure 5.

```
# sending command "movej" to the robot driver, space of joints
def moveToConfiguration(self, q, a=0.5, v=0.5, t=0, r=0):
    self.sendURScript("movej({}, a={}, v={}, t={}, r={})\n".format(q, a, v, t, r))
    return True
```

Figure. 5. Movement command generator.

Tablet pen was attached to the robot arm by the holder developed in the work (Figure 1). One of the five results of robot writing experiment is presented in Figure 6. Acceptable duration and accuracy parameters of the robot writing process were selected experimentally. Longer duration of the writing process can provide the better robot writing accuracy.

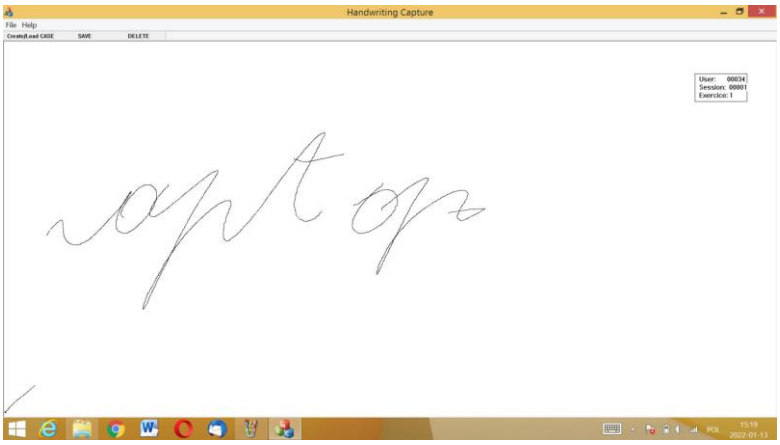


Figure 6. One of the results of robot writing – screenshot of WacomGUI2 program.

5. Conclusions

The novel human-robot interaction method presented in the paper allowed performing the robot writing task to mimic human writing. The connection between human-operator and UR3 robot has been established via Wacom graphic tablet, Polyscope panel, and control computer (laptop) (Figure 1). To implement robot-computer interactions a new data frame was developed using ROS package and Python functions. The data received from UR3 robot were converted and interpreted on the computer by Python communication script developed. Our proposed computer-robot communication method allowed implementation of the human-robot interaction in the robot writing process. At the same time, the method proposed assumed the robot pen motion from point to point. Therefore, it results in the robot arm vibrations making the writing process imperfect. Therefore, few program modules created in this study to perform robot writing have to be improved to provide higher robot writing accuracy. The best writing results were obtained with acceleration 0,6, speed 0,15, time 0, and blend 0 parameters of *movej* command. These parameters provided optimal robot writing for the proposed method.

The computer-robot communication method developed can be implemented in various practical tasks where Universal Robots and other manipulators are used (Quintana & al., 2018). With a few minor modification, the developed program units can be used for control and communication with UR5, UR10 and other robots. After additional tests and experiments, the created Python program modules for robot writing may make robot arm more human-like in the writing process. Information about time and tilt of robot pen can be further used to develop a more smooth and accurate robot writing process.

References

Miatliuk, K., Wolniakowski, A., Diaz, M., Ferrer M.A. & Quintana J, (2019). Universal Robot Employment to Mimic Human Writing, 20th Int. Carpathian Control Conf., *Proc. of the IEEE*, New York, 1-6.

Wolniakowski, A., Diaz, M., Quintana, J., Miatliuk, K. & Ferrer M.A. (2021). Towards human-like kinematics in industrial robotic arms: a case study on a UR3 robot, 54th IEEE International Carnahan Conference on Security Technology, *IEEE xplore*, 1-5.

WacomGUI2 (2021), http;//github.com/roure/handWritingCaptureWacom

UR3 user manual (2022). https://www.universal-robots.com/

RoboDK simulator for industrial robots and offline programming (2022). https://robodk.com

Python programming language (2022) https://www.python.org

ROS operating system (2021), www.ros.org

URScript Programming Language, Universal Robot Manual (2021). http://www.sysaxes.com/manuels/

Wolniakowski, A. (2014). GITLab. https://gitlab.com/dagothar

Kordziukiewicz, T. (2022). Realization of writing processes by universal robot. Bachelor thesis, ed. K.Miatliuk, BUT, Bialystok, Poland.

Quintana, J.J., Diaz, M. & Ferrer M.A. (2018). Stirring up the Learning to Program Robotic Arms Through the Generation of Student Handwriting. 13th Conference on Technologies Applied to Electronic Teaching, Spain, 20-22 June 2018, 1-6.