ME/SE 740 final report - Robot Hand, Human Gesture Imitation

Lingyan Jiang

April 30, 2020

Abstract

In this project, I want to combine what I learned in the class about vision, robotics and pose-based (PBVS) visual servoing to let the robot hand copy simple human hand gestures and motions such as bending a single finger or making a fist. I assembled a robot hand with six degrees of freedom, five mimic fingers with six servos to control and used Lobot Servo Control Software to Conduct kinematics and adjust deviation. An "eye to hand" single video Pi camera was used to capture the image and analyze it by raspberry pi via Python, AipBodyAnalysis and Opencv. By analyzing five fingers and the human arm in the image, it will conduct the command to move the robot hand.

1 Introduction

Robotic hands have always been the spotlight of popular technology. Especially the biomimetic anthropomorphic robot hands that use EMG sensors to analyze muscle electrical activity to control a robot hand with pressure sensors to relay touch sensations have been the focus of IEEE Spectrum and (successfully built by Zhe Xu *at* Yale University's GRAB Lab [1]). Most robot hands use flex sensors which link human hand and robot hand and correspond to the position of the server via reading codes of analog value and maps. Another part of the study uses image-based visual servoing to control the motion of the robot system. In this context, the visual system can be integrated by single or multiple cameras, mounted in fixed (eye-to-hand) or moving (eye-in-hand) configurations. The visual servoing techniques can be widely classified into the following categories: position-based, image-based, or hybrid, which combines the common characteristics of the first two [2].

Robotic hands gesture recognition and imitation is in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Many approaches have been made using cameras and computer vision algorithms to interpret sign language [6]. Depending on the type of the input data, the approach for interpreting a gesture can be done in different ways. However, most of the techniques rely on key pointers represented in a 3D coordinate system. Based on the relative motion of these, the gesture can be detected with high accuracy, depending on the quality of the input and the algorithm's approach. The scientific

literature differentiates two different approaches in gesture recognition: a 3D model based and an appearance-based approach.

3D model-based algorithms can use volumetric or skeletal models, or even a combination of the two. Volumetric approaches have been heavily used in the computer animation industry and for computer vision purposes. The models are generally created from complicated 3D surfaces, like NURBS or polygon meshes. The drawback of this method is that it is very computational intensive, and systems for real time analysis are still to be developed. Instead of using intensive processing of the 3D models and dealing with a lot of parameters, skeletal-based algorithms can just use a simplified version of joint angle parameters along with segment lengths. Appearance-based models no longer use a spatial representation of the body, because they derive the parameters directly from the images or videos using a template database. Some are based on the deformable 2D templates of the human parts of the body, particularly hands. These template-based models are mostly used for hand-tracking, but could also be of use for simple gesture classification [6].

Learning to imitate and copy human gestures can help society in many aspects, such as reading sign language, from imitating gestures to outputting the meaning of language, and even combining voice language libraries to allow gestures to "speak". Moreover, the simulation of subconscious behavior of humans can help design future robots. It also understands the limitations of human hands. For example, when the little finger is bent, the ring finger is also bent subconsciously, which also strengthens the simulation of the robot hand. It can even build an intuitive human-robot interaction (HRI) framework for gesture and human behavior recognition. It relies on a vision-based system as interaction technology to classify gestures and a 3-axis accelerometer for behavior classification (stand, walking, etc.). An intelligent system integrates static gesture recognition recurring to artificial neural networks (ANNs) and dynamic gesture recognition using hidden Markov models (HMM)[3].

2 Technical Approach

To achieve a robot hand that imitates human gestures, I roughly divide the technology approaches into four steps: assembling the robot hand, debugging deviations and programming to control the movement of the robot hand, assembling the raspberry pi and Pi camera for capture image of human hand , and finally using raspberry pi for image-based analysis and executing the code to move the robot hand.

Assembling the robot hand

The first step of my project is to assemble a robot hand with six degrees of freedom, five mimic fingers. I bought the mimic finger parts, six digital servos, one controller, several U-shaped brackets, bearings, screws, pillars and circuits online. I assembled all the parts together which is shown in Figure 1. As we can see in the figures that the black parts at each joint of the hand are the digital servos controlling the freedom of the robot hand.

The freedom and range of different positions of the robot hand were controlled by six servos from bottom to top. The bottom servo is below the bottom station plate, which imitates the rotation of the human arm. Servo cooperates with bearings to turn the robot arm 90 degrees to the left and right. Because the rotation range of the human arm is about the same, there are no restrictions when debugging. The two servos above control the robot arm flipping up and down. The amplitude range is 180 degrees. Here it is imitating the rotation of the human arm up and down. Above it is a horizontal servo, which is used to imitate the up and down movement of the human palm, and the range is also 180 degrees. Because the range of ups and downs of the human palm is about 110 degrees, it will be reduced to 110 degrees during debugging deviations. The second servo from the top is imitating the left and right swing of the human wrist. The range is also 90 degrees left and right, because the human wrist swings about the same range, so there is no amplitude deviation adjustment. The last servo is in the top position, assembled in the palm of the robot hand. This servo is used to control the extension and contraction of five fingers, about 90 degrees.

After the robot hand part is assembled, I need to connect the robot hand to the control board. In Figure 2, it is a detailed picture of the control board. At the top of the control board are the digital servo interfaces. There are more than twenty digital servo interfaces in this control board. I used the 10th to 15th interfaces because of the convenient connection. Computer data interface is for computer control and command. In addition, there is an MCU communication interface, battery interface, buzzer, communication signal light and power switch on the control board.

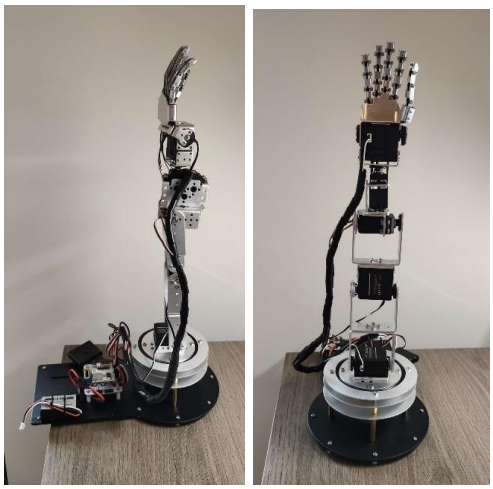


Figure 1: These two pictures are the front and side views of the assembled robot hand. The black parts at each joint of the hand are the digital servos controlling the freedom of the robot hand.

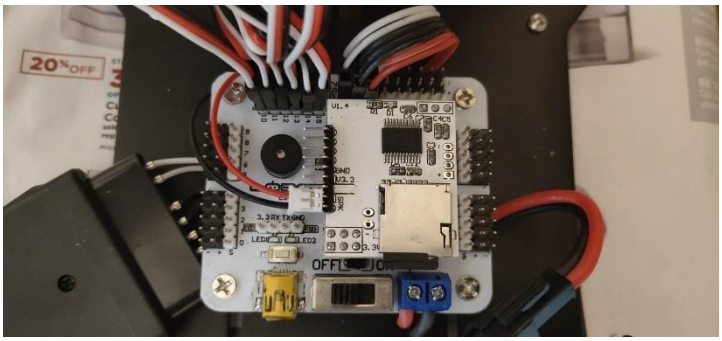


Figure 2: The control board that connected to the robot hand.

Conducting kinematics, programing and adjusting deviation — Lobot Servo Control Software

After the robot hand is assembled, the robot hand needs to adjust the deviation, limit the servo range and program the motion command. Here I use Lobot Servo Control Software, as shown in Figure 3 below. First, I select the servos to be controlled. Here I use the 10th to 25th servo interface, so I choose the operation interface No. 10 to 15. When the robot hand is connected to the computer, the red light at the upper right will turn to green. First, I reset the servo to return the servo to the out-of-state mode, and then use the buttons on the operation interface to move left or right to adjust the servo state back or forth. After the servo is adjusted to the horizontal or vertical state, save the deviation, and then I can start programming the robot hand movement operation. The initial state of the robot hand servo is set to 1500, the left and right adjustment value can make the robot hand servos move, and the time adjustment below can control the movement time of the servos. For example, the initial state can be set as # 10P1500 # 11P1500 # 12P1500 # 13P1500 # 14P1500 # 15P1500. Changing the servo value on the computer can change different movements of the robot hand. For this, I can edit the movements and perform motion operations on the robot hand on the computer.

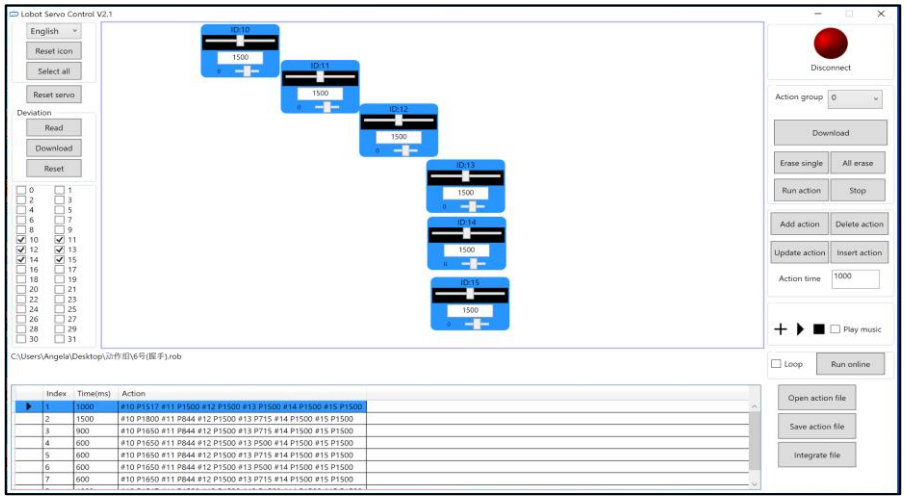


Figure 3: Lobot Servo Control Software operation interface to program the motion command and adjust deviation

Assembling an “eye to hand” camera and capturing image

I ordered a raspberry pi to link with the robot hand as a control. I use one "eye to hand" single video Pi camera which can connect with the raspberry pi, as shown in Figure 4 below. The camera is facing the palm of the human hand and the same direction as the robot hand. I set the camera above the raspberry pi to facilitate the connection of the data cable. In order to capture the image, I use python to program in raspberry pi. The palm of the tester is horizontally opposed to the palm of the robot at a certain distance. The tester performs simple basic hand gestures in front of the camera, and the image captured by the camera. I choose a white wall as the background of the image to capture due to the low resolution of the camera. As shown in Figure 5, these are the codes I use to capture the images with a gap of 5 seconds. Sleep function is used to delay the time of capturing images and can better help the tester make gestures and keep the gesture stable. As we see in Figure 6, two different images of human hand movement are taken and stored in raspberry pi for later analysis.

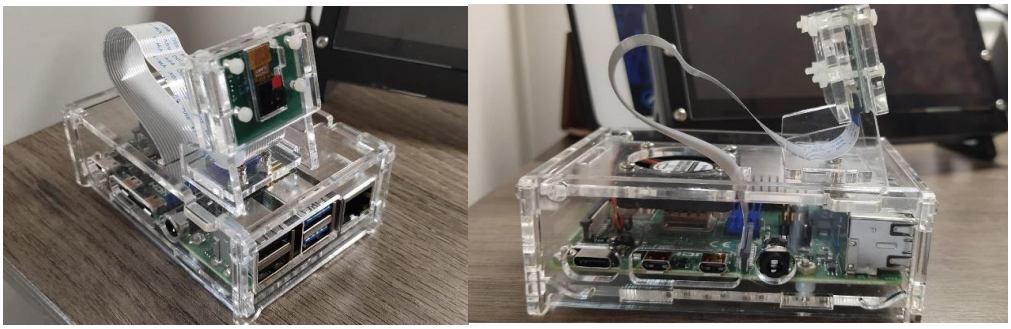


Figure 4: The camera for capturing the image of the human hand and connecting with the raspberry pi.

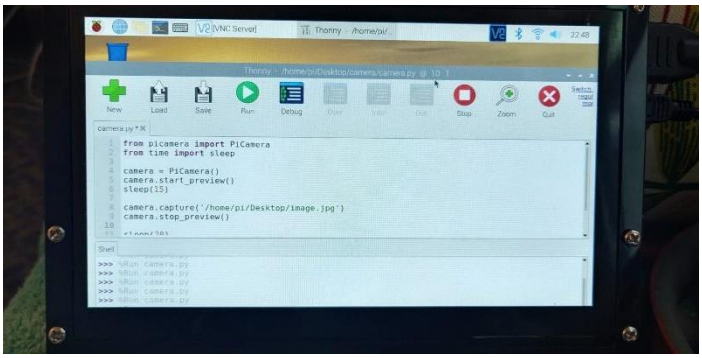


Figure 5: The code to capture the images.

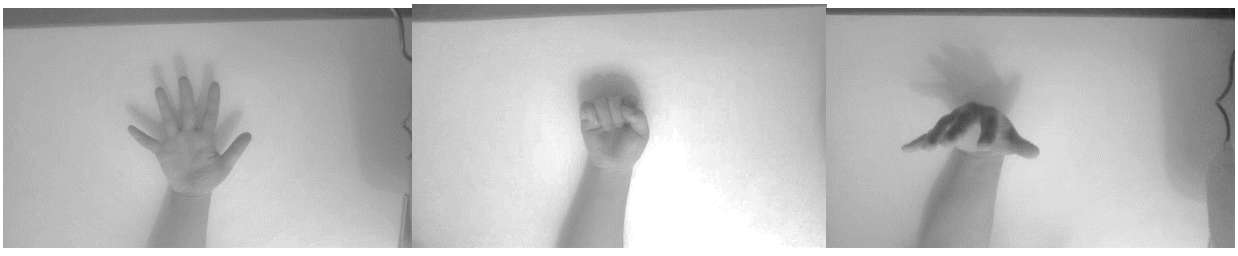
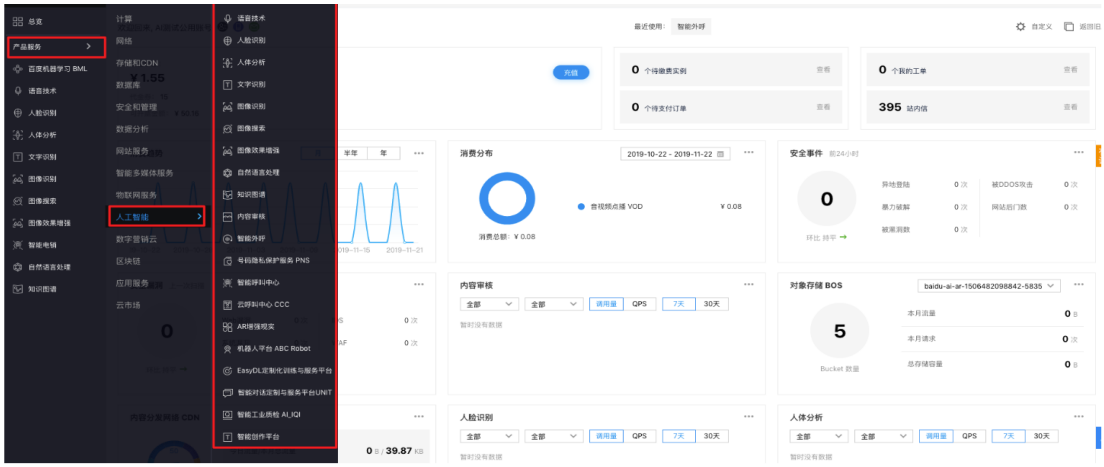


Figure 6: Three different human hand movement images captured by the camera.

Images analyzed by raspberry pi via python and conduct command

For this step, I use Python to analyze the images to recognize gestures with Baidu AIP, AipBodyAnalysis and Opencv2 and conduct the command via GPIO command. Baidu AIP is an online platform based on the world's leading Baidu Brain Corporation, Tianzhi artificial Intelligence provides universal AI capabilities, industry solutions, AI development platforms, and training services such as ready-to-use vision and voice and it is free for students to use. People who use Baidu AIP can customize personalized smart applications based on their business needs and data. From using its database and Downloading their SDK python, C, java documents, it's easy for us to use in speech recognition or synthesis, face or body recognition, natural language processing, text recognition, image processing, AI platform development, AR technology development, etc. There is also one shortcoming, since it is developed by Baidu Corporation which is a Chinese company, so most output of the result is in Chinese. As we see in figure 7, it shows the website surface of Baidu AIP and AipBodyAnalysis. In Figure 8, this is the code I use and the output of gesture recognition results. After the raspberry pi recognizes the gesture, I use GPIO to conduct the command on the servos via python. The control of the servos generally requires a time-base pulse of about 20ms. The high-level part of the pulse is generally the angle control pulse part in the range of 0.5ms-2.5ms, and the total interval is 2ms. Taking 180 degree angle servo as an example, control relationship is shown in Figure 9.

Before the Python instruction starts, I let the initial robot hand keep open and vertical, and each time the movement state is changed, it returns to the initial state, which is used to more accurately coordinate the individual fingers or joints. At first, I will start by showing a basic open palm in front of the camera and after the first image is captured, I will change another gesture as making a fist. After completing the first step, I use the length of the finger and the proportion of missing fingers to recognize basic gestures[4]. After the raspberry pi analyzes five fingers in the image completely, it will conduct the command to open the robot hand and make the action as making a fist.



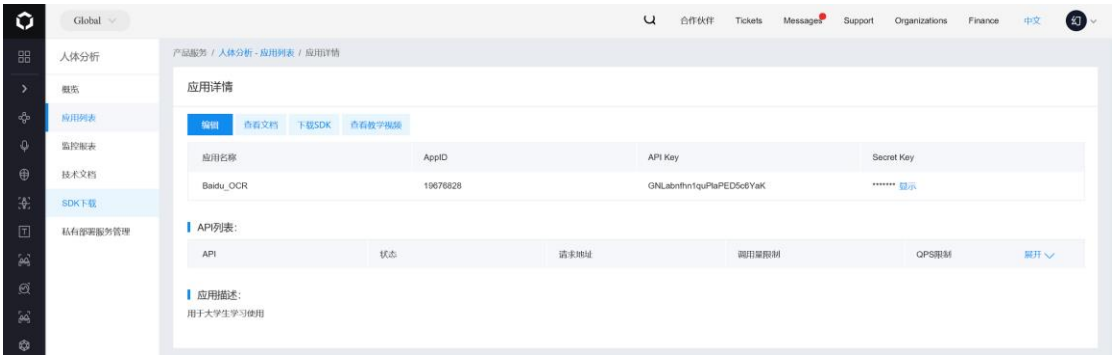


Figure 7: The online AI platform of Baidu AIP and AipBodyAnalysis website surface.

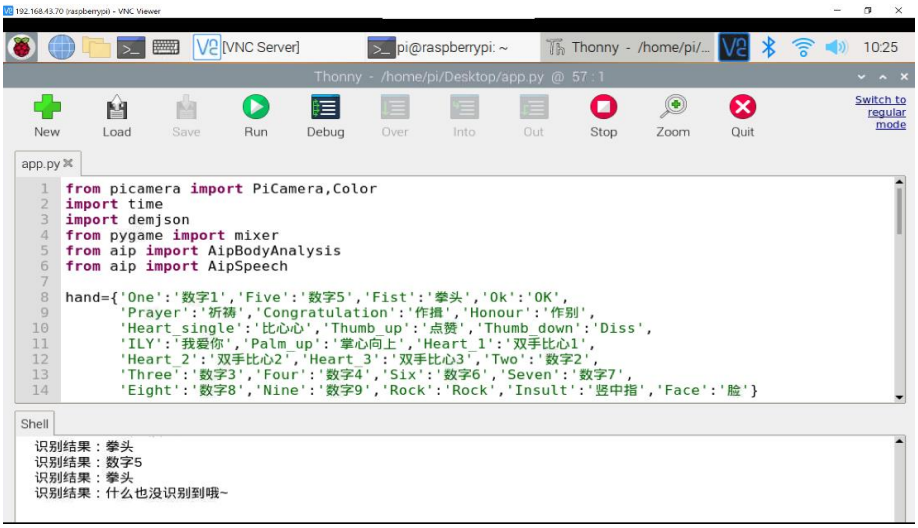


Figure 8: The code to recognize the human gesture and conduct a command to control the servos to make actions such as making a fist.

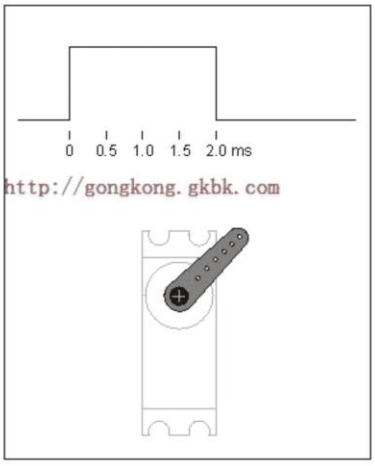


Figure 9: The control of servo corresponds to the pulse in the range of 0.5ms-2.5ms. This figure is from http://gongkong.gkbk.com.

3 Results - limitations and improvements

Although I completed the preliminary project, there are still many limitations and areas for improvement.

Limitations

1. When assembling the robot hand, because the palm of the robot hand has insufficient space, it is not possible to add a servo to control the bending or stretching of each finger. So five fingers can only be bent or straightened at the same time, and there is no single operation.

2. When capturing images, because the resolution of the camera is very poor, the requirements on the background of human hands are very high. I used a pure white wall as a background to take photos for analysis. If an image with a non-white background is used, it is very easy to report an error and the code command cannot be executed.

3. I did not use a video recording function in this project. Although a raspberry pi camera can record and analyze, it takes too long. Therefore, I only use the sleep function for batch shooting. When capturing the image, the human hand of the tester needs to maintain the posture and be still, so that it can be compared with the previous images.

4. When installing the camera, it was originally supposed to be installed in the palm, which is more convenient for capturing images of human hand gestures. However, since a servo has been installed here, there is no extra space to install the camera. In addition, the link line between the camera and raspberry pi is not long enough, so it can only be set to “eye to hand” camera, and the camera is placed above the raspberry pi.

5. When analyzing images, I can not actually analyze whether the human hand of the tester has turned the arm to the right or the left or not. Because, unlike bending arms up and down, the gap between turning arms right or left is very subtle, and there are no missing fingers or arms. This requires a highly accurate algorithm to analyze.

Improvements

1. I will improve the space in the palm of the robot hand to add a servo. This new adding servo can control each finger to bend or stretch.

2. I will use a camera with a higher resolution. Secondly, it needs to be improved in the process of python image analysis. I will try it with different algorithms.

3. I will try to use the camera function to facilitate the recording of human hand gesture movements. I can also try the LSTM neural network algorithm to analyze the human hand movement video.

4. In the future, I will use a longer camera link line and add more space between the palm and fingers of the robot hand to set up an “eye in hand” camera. In this way, with two cameras, it can better capture the movement of human hands from different perspectives.

4 Conclusion and future

For this project, I only completed the basic part and the command executed on the robot hand is also basic. Through the above four steps, the human gestures that can be imitated at present are just fist making and wrist rotation, the two most basic human hand gestures. If I want to completely imitate human gestures, there is still a long way to go. In the last paragraph, I detailed the shortcomings of this project and the areas where it can be improved. After this semester, I will improve these deficiencies and make corrections. For the future of this project, I will further improve the camera type, number and location, 3D image analysis, error estimate of finger coordinates, kinematic calibration, etc. For example, I can use an infrared vision system, which also converts images in hand shape features, the Leap Motion Controller (LMC). In this way, a 3D image can be constructed for analysis, kinematic calibration can be better performed, and the range and degree of accurate finger movement can be accurate. I may also use 3D print to mimic anthropomorphic fingers and joints to mimic human skeleton hands. I may even add EMG sensors attached to the arm to analyze muscle electrical activity to control a robot hand which combines the two most popular areas of robot hand approaches in the future.

5 Video and codes

For the video and codes, I have uploaded them on my youtube and github and the links are listed as below.

Conduct kinematics, program and adjust deviation: https://youtu.be/PpB0mopm6A8

Robot gesture imitating result: https://youtu.be/wKlsRg4rpp8

Github link of the code: https://github.com/lingyanj/Robot-hand-gesture-imitating

6 References

[1] Ackerman, E. (Ed.). (2016, February 18). This Is the Most Amazing Biomimetic Anthropomorphic Robot Hand We've Ever Seen. Retrieved April 29, 2020, from https://spectrum.ieee.org/automaton/robotics/medical-robots/biomimetic-anthropomorphic-robot-hand

[2] Reis, M., Leite, A., From, P., Hsu, L., & Lizarralde, F. (2015). Visual Servoing for Object Manipulation with a Multifingered Robot HandThis work was partially supported by the Brazilian Funding Agencies (CNPq, CAPES and FAPERJ) and The Research Council of Norway. IFAC PapersOnLine, 48(19), 1-6.

[3] Mendes, N., Ferrer, Vitorino, Safeea, & Neto. (2017). Human Behavior and Hand Gesture Classification for Smart Human-robot Interaction. Procedia Manufacturing, 11, 91-98.

[4] Raheja, J., Shyam, R., Kumar, U., & Prasad, P. (2010). Real-Time Robotic Hand Control Using Hand Gestures. 2010 Second International Conference on Machine Learning and Computing, 12-16.

[5] 7inch HDMI Display-C. (n.d.). Retrieved April 29, 2020, from http://www.lcdwiki.com/7inch_HDMI_Display-C

[6] Gesture recognition. (n.d.). Retrieved April 30, 2020, from https://en.wikipedia.org/wiki/Gesture_recognition#Types_of_touchless_technology