Visual Control Robotic Arm and Force Tactile Feedback of Gripper

Jia-jun Long, Yi-jun Fang, Yi-fei Chen, Ke-run Xiang, and Zi-meng Wang
Department of Mechanical and Energy Engineering
The Southern University of Science and Technology, Shenzhen, China

Abstract–This paper introduces the invention of the 5-DOFs robot arm's architecture, visual control by Python, calculation of forward and inverse kinematics and work, procedure of trajectory planning and MATLAB simulation, operation of pressure sensor and the future solution for the visual control robot arm. Although our team didn't work on this topic before, we decided to study and explore more contents which will be covered in this article.

***Index Terms*—MATLAB simulation, pressure sensor, trajectory planning, visual control, forward kinematics**

I. Introduction

NOWADAYS the application of robotic arm in production and life has been more and more extensive. With its high precision and low operating cost, industrial robot arm gradually replaces manual operation, and is mainly used in petrochemical, textile, steel and other large production volume and harsh production environment. It is a trend of contemporary times that robotic arm is integrated into every aspect of production and life. Research on visually controlled robotic arms is also under way.

At present, the mechanical arm is mainly used in the industrial field, but it is seldom used in daily life. In this respect, our project can do:

1. Remote control

In case of force majeure, such as the outbreak of epidemic, distance and objective natural conditions greatly hinder our work development. Our mechanical arm can get rid of space constraints and complete remote tasks by mapping human movements.

2. Fix AI issues

AI remote control cannot reproduce the power and experience exerted by human workers, and force-haptic control can accurately complete some delicate operations.

3. VR connection

The development of virtual reality and 5G has opened up a broader use of VR equipment, which was originally entertainment, to recognize human posture through visual recognition, so as to realize the interaction between virtual world and real world. Create control modes for VR+ robots. The control and drive of a single robot arm or manipulator can be applied to more human-like robots in the future, and even create VR controlled robot doppleganger.

mds
August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here

Subsubsection text here.

Manuscript received December 1, 2012; revised August 26, 2015. Corresponding author: M. Shell (email: http://www.michaelshell.org/contact.html).

II. Experiment Methods

A. Mechanism Design and Assembly

First, we roughly constructed a five-axis robotic arm based on the human arm to facilitate the subsequent mapping of the human arm joints. Combining the parts of the given kit and the parts matching tutorial attached with the kit manual and the assembly steps of some robots provided by ROBOTIS official website, and considering the performance of the motor, we designed a motor parallel manipulator and completed the parts of the main part of the install of the robotic arm.

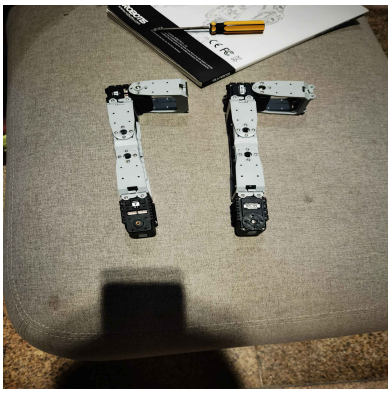

Fig. 1. The arm

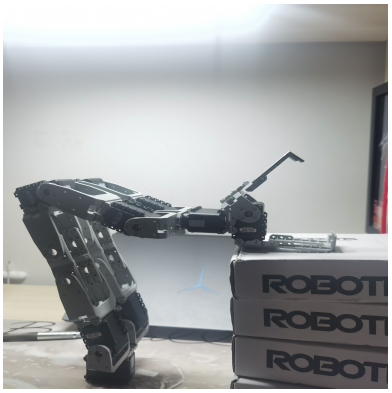

Fig. 2. The robotic arm

Since the parts of the given kit cannot fully satisfy our fixation of the structure of the robotic arm part, we bought a double-headed nut to connect and fix the two parallel robotic arm parts.

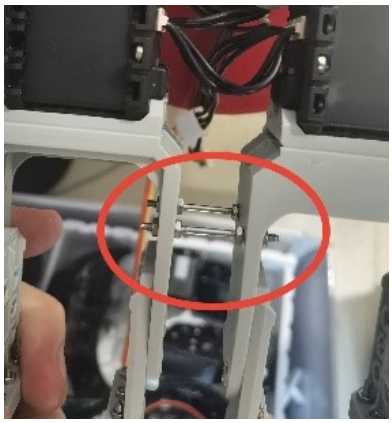

Fig. 3. The materials

Fig. 4. How to use

In order to obtain the complete modeling of the robotic arm and the quality data of each joint and reduce the part error so that the 3D printed parts and the kit parts can be well matched. We found the step files of all parts on the official website, and successfully converted the step files to solidworks part format through a file conversion website, realizing almost zero error for all parts.

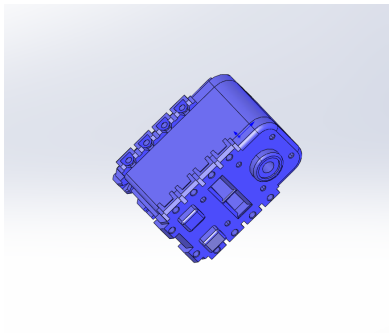

Fig. 5. Electrical Machinery

Combined with our part size, we designed a turntable to achieve the fixation of the robotic arm and the rotation of the uppermost joint. And assembled with the main body of the robot arm.

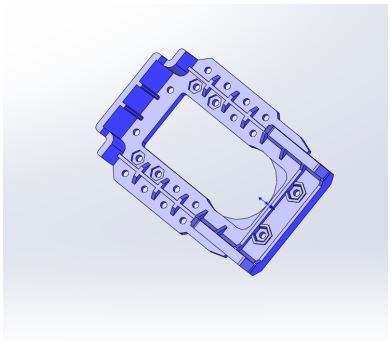

Fig. 6. Electric frame

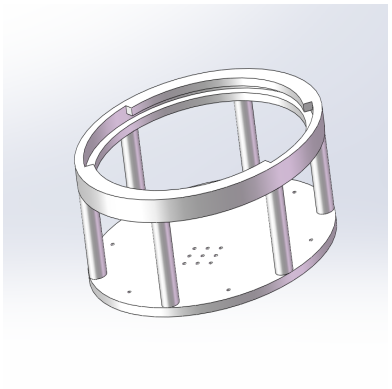

Fig. 7. Pedestal

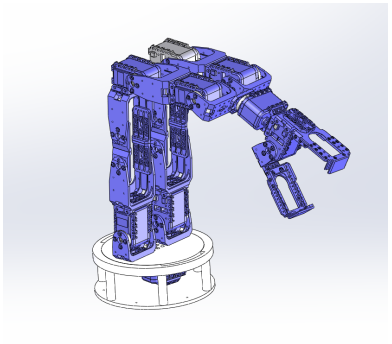

Fig. 8. The pressure sensor install1

Through 3D printing, we made the designed turntable and combined it with the robotic arm. Since the wiring length given by the kit is limited, we planned the wiring through the extension cords purchased by ourselves to ensure that each joint can be rotated to the greatest extent without being limited by the wiring length.

Since the clamping jaws installed with the parts of the kit are relatively simple, and the size of the objects to be grasped is relatively high, we have added sponges and used the concavities of the sponges to adapt to a wider variety of object sizes. At the same time, our pressure sensor needs to be fixed on a certain force surface, and the original set of jaws has no flat force surface, which cannot meet the installation requirements of the pressure sensor. Therefore, after adding a sponge, it is shallower from the outer surface of the sponge.

Fig. 9. The pressure sensor install2

Dig a hole to integrate the pressure sensor in. Although the pressure sensor is not in direct contact with a hard surface, we can still roughly measure the force of the gripper based on the force transmission.

Fig. 10. The base effect

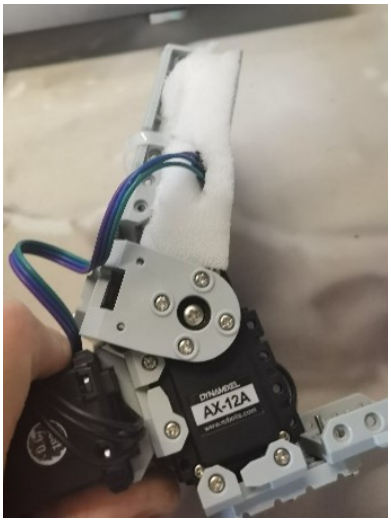

Fig. 11. The base effect

Finally, we completed the construction of the complete model of the robotic arm.

Fig. 12. The final robot arm

B. Visual Control and Implementation

Considering that Python is quite mature in terms of visual recognition, Opencv of Python is used for visual reading and analysis. However, our final goal is to realize the mechanical arm in space, which requires the coordinates of each joint on the arm. However, considering that the laptop camera can only read the coordinates of a specific point on the plane, we need to get the three-dimensional coordinates of a specific point from continuous plane images.

After surfing the Internet, we decided to use Mediapipe, an open source application framework for multimedia machine learning models developed by Google Research. Meidapipe takes real-world 3D training data, uses AR synthetic data generation, uses ML pipes for 3D object detection, and places virtual objects into scenes with AR session data. This allows us to generate physically possible positions using camera posture, detected planes, and estimated lighting. And use lighting that matches the scene.

In addition, compared to expensive 3D cameras, Mediapipe uses a more convenient solution, requires less camera, and is more in line with our lightweight design.

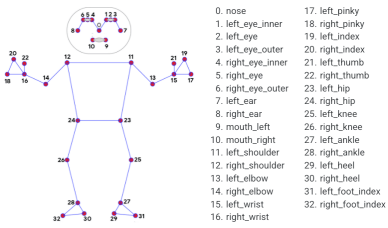

Fig. 13. The model

This is the official definition given by mediapipe to invoke various joints in the Holistic part of human body. What our team needs to establish is a five-axis mechanical arm. Here, we choose to use the right arm of human body to read data and output points 12,16,18,20,22,24 and so on. We can get the spatial coordinates of the corresponding angles, and use forward kinematics to control the manipulator.

We define a poseDetector class to encapsulate the call to MP:

It defines various methods for calling Mediapipe:

First, we define the $findPose()$ function

First, we will create a copy of the original image so that nothing of the original image is lost during preprocessing.

We will convert the blue-green-red format to red-green-blue format later because it is widely used in computer vision.

We will now use the Process function to handle posture detection for the transformed image format.

It is time to do some validation and check if landmarks are detected in the image. If landmarks are detected, we will use the draw_landmarks function to draw them on the image.

Previously we first validated before drawing the landmark, now we will check whether the display parameters allow us to display the results and input images.

If the above condition is true, the original image is displayed along with the generated image. Otherwise, it just returns the output image and results.

Second, we define a function to $findPosition()$

Build an array to store the coordinates of key nodes

Check whether self.results.pose_landmarks is empty,

Use the for loop to loop through self.results.pose_landmarks in enumerate

Get the ids of each joint and their positions and append them to the lmList.

At the same time, in order to better output data, we encapsulated the output method:

During initialization we modulate mediapipe's various parameters:

Static_image_mode: Here we need to stream video, so we set the value to False.

min_detection_confidence and min_tracking_confidence are both set to 0.5, and confidence levels are set for the detection and prediction models, which are also the initial values given on the official website.

min_tracking_confidence This is the confidence of the tracking level, that is, it will detect whether people are detected according to the confidence provided by us, so that the problem of high robustness will not occur after detection, and the stability of the system will be improved. This confidence level is ignored when static_image_mode is True.

Considering that MATLAB has a part of the function can not be implemented in Python, and through the toolkit call, can make the code more efficient and clear. We need to transmit the obtained data between Pycharm and MATLAB. Finally, after the output file is selected, MATLAB reads the data again for data transmission.

Code flow:

First we instantiate VideoCapture and set up to read the video stream directly from the camera or read the file from the file according to our needs. Then we instantiated the poseDetector and initialized all the confidence levels. After reading the image, we used our instantiated detector to call $findPose()$ method to get each joint of human body. Then call $findPosition()$ to get the three-dimensional coordinates of each key in space, then we output according to the required, and provide it to MATLAB for analysis and subsequent control.

Run into difficulties:

When debugging we also encountered many difficulties, such as the determination of coordinate system for mediapipe itself coordinate system is set among human hip to the origin, but go ahead with output when we want to coordinate system into the world, but for x axis and y axis, there are certain the size of the magnification, For the Z-axis of the connection between the camera and the human body, there was no distance calibration. Finally, we considered that direct calculation of the Angle would have no influence on which coordinate system the coordinates belonged to, so we finally decided to directly output the spatial coordinates of the coordinate system under the camera.

Fig. 14. Images obtained during testing

In addition, when encapsulating classes, I learned Python code specifications from the Internet and configured OpencV with Anaconda, which can be equipped with a virtual environment, because I had never been exposed to Python syntax. Only after many mistakes were made, our visual reading had almost no error.

Moreover, Since the results of visual recognition are all the world coordinates relative to the camera, when the human body keeps the same movement, the depth prediction will be affected by external interference factors and will change to a certain extent. Due to the size difference between the human arm and the mechanical arm, the conversion coefficient calculated by us is only based on our theoretical calculation. So a small change in the depth of recognition is likely to result in a large movement of the arm. We found these movements by testing flat motion, such as bending the elbows and wrists. When we test the stereo position, such as the rotation of the shoulder and the rotation of the forearm, the deflection of the manipulator itself will increase.

Our human posture recognition is carried out using python, while the control is using MATLAB. Because of the incompatibility of the software, the way of real-time data transmission is very rough. Real-time reading of python exported data by Matlab will cause a delay in the control of the manipulator, which will behave badly when running 4min.

C. MATLAB Simulation

1) Import of Robotic Arm

In this part, we will introduce how we use the MATLAB-robotic-toolbox made by Peter Cork to do the simulation. First,

we use MATLAB to establish a five-axis manipulator model and add dynamics parameters to the manipulator model. All the parameters were read from the model from SolidWorks. These parameters include the mass of each joint, the position of the center of mass, and the inertia tensor of each joint. Then the simulation of robotic arm is ready. These parameters will provide great convenience and help for Lagrangian dynamics calculation later.

2) Kinetic Analysis

In this part, we conducted DH convention analysis on the basis of the five-axis manipulator model established. In setting DH parameters, we encountered some difficulties. For example, under the definition of DH convention, the coordinate system of joint 4 could not be directly connected with that of joint 5, resulting in no solution. After collecting relevant information on the Internet, it is found that such a situation can be solved by setting joint 4 and joint 5 at the same coordinate origin and putting the length of the manipulator in the latter DH for output.

The following figure shows our DH parameter table and schematic diagram of the manipulator:

Joint	θ_i	d_i	a_i	α_i
1	θ_1^*	l_1	0	-π/2
2	θ_2^*	0	l_2	0
3	θ_3^*	0	l_3	0
4	θ_4^*	0	0	π/2
5	θ_5^*	l_5	0	0

Fig. 15. DH Convention

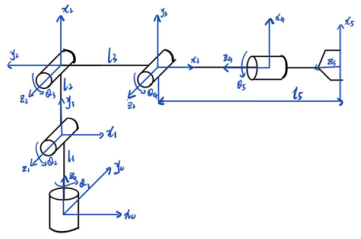

Fig. 16. Schematic diagram of robot arm

Secondly, on the basis of the problem of the manipulator model, we carry out the kinematics analysis of the manipulator.

Let's first analyze the forward kinematics.

The transformation matrix of each joint was obtained by DH parameters. Based on the 3d coordinates of each joint obtained in Python, I made angle conversion through MATLAB to obtain the change angle of each joint of the human arm. Then the angle is substituted into the transformation matrix to obtain the end-effector pose. The specific process is shown below:

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & l_2 \cdot c_2 \\ s_2 & c_2 & 0 & l_2 \cdot s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & l_3 \cdot c_3 \\ s_3 & c_3 & 0 & l_3 \cdot s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_2 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then we calculate the end-effector pose:

$T_5 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5$

$$T_5 = \begin{bmatrix} \delta_{11} & \delta_{12} & \delta_{13} & \delta_{14} \\ \delta_{21} & \delta_{22} & \delta_{23} & \delta_{24} \\ \delta_{31} & \delta_{32} & \delta_{33} & \delta_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$\delta_{11} = c_1 \cdot c_{234} + s_1 \cdot s_5$
$\delta_{12} = -c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5$
$\delta_{13} = -c_1 \cdot s_{234}$
$\delta_{14} = c_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_23 + l_2 \cdot c_2$
$\delta_{21} = c_1 \cdot c_{234} - s_1 \cdot s_5$
$\delta_{22} = -s_1 \cdot c_{234} \cdot s_5 + c_1 \cdot c_5$
$\delta_{23} = -s_1 \cdot s_{234}$
$\delta_{24} = s_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2$
$\delta_{31} = -s_{234} \cdot c_5$
$\delta_{32} = s_{234} \cdot s_5$
$\delta_{33} = -c_{234}$
$\delta_{34} = l_1 - l_2 \cdot s_2 - l_3 \cdot s_{23} - l_5 \cdot c_{234}$

Gripper's position is defined by vector p_5 while orientation defined by x_5, y_5, z_5provided by the forward kinematics[1]:

$$p_5 = \begin{bmatrix} c_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2 \\ s_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2 \\ l_1 - l_2 \cdot s_2 - l_3 \cdot c_{23} - l_5 \cdot c_{234} \end{bmatrix}$$

$$x_5 = \begin{bmatrix} c_1 \cdot c_{234} \cdot c_5 + s_1 \cdot s_5 \\ s_1 \cdot c_{234} \cdot c_5 + c_1 \cdot s_5 \\ -s_{234} \cdot c_5 \end{bmatrix}$$

$$y_5 = \begin{bmatrix} c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5 \\ -s_1 \cdot c_{234} \cdot s_5 + c_1 \cdot c_5 \\ s_{234} \cdot s_5 \end{bmatrix}$$

$$z_5 = \begin{bmatrix} -c_1 \cdot s_{234} \\ -s_1 \cdot s_{234} \\ c_{234} \end{bmatrix}$$

And then we're going to do the inverse kinematics. Solution to the inverse kinematics problem is reduced to search of the arguments q_1, q_2, q_3, q_4, q_5 based on the gripper's position and orientation. The equations above can be rewritten in the form:

$x_{5x} = c_1 \cdot c_{234} \cdot c_5 + s_1 \cdot s_5$
$x_{5y} = s_1 \cdot c_{234} \cdot c_5 + c_1 \cdot s_5$
$x_{5z} = -s_{234} \cdot c_5$

$$y_{5x} = -c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5$$
$$y_{5y} = -s_1 \cdot c_{234} \cdot s_5 - s_1 \cdot c_5$$
$$y_{5z} = s_{234} \cdot s_5$$
$$z_{5x} = -c_1 \cdot s_{234}$$
$$z_{5y} = -s_1 \cdot s_{234}$$
$$z_{5z} = c_{234}$$
$$p_{5x} = c_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2$$
$$p_{5y} = s_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2$$
$$p_{5z} = l_1 - l_2 \cdot s_2 - l_3 \cdot c_{23} - l_5 \cdot c_{234}$$

Based on the equations above, we can easily get the position of each joint:

$$i = l_1 - l_5 \cdot c_{234} - p_{5z}$$
$$j = p_{5x} \cdot c_1 + p_{5y} \cdot s_1 + l_5 \cdot s_{234}$$

$$q_1 = \arctan(\frac{p_{5y}}{p_{5x}})$$

$$q_2 = \arctan(\frac{i \cdot (l_2 + l_3 \cdot c_3) - j \cdot l_3 \cdot s_3}{i \cdot l_3 \cdot s_3 + j \cdot (l_2 + l_3 \cdot c_3)})$$

$$q_3 = \arccos(\frac{i^2 + j^2 - {l_2}^2 - {l_3}^2}{2 \cdot l_2 \cdot l_3})$$

$$q_4 = 0 - q_2 - q_3$$

$$q_5 = c_{234} \cdot q_1 - 2 \cdot \arctan(\frac{x_{5y}}{x_{5x}})$$

From above, we get the rotation angle of each joint, and we can calculate the angular velocity and angular acceleration at the moment by the angle, so the forward and inverse kinematics calculation has been basically completed.

However, in the process of forward and inverse kinematics, we have encountered many difficulties and tried many methods to calculate the kinematics.

In the use of the most convenient MATLAB robotic toolbox, its own $fkine()$ and $ikine()$ two functions, can be very convenient to solve the forward kinematics and inverse kinematics. However, in the experimental process, we found that it has limitations. In some special positions, the array will be out of bounds error, and this function is extremely difficult to check and repair, so we found other methods on the Internet. As we ended up using, we calculated directly from DH parameters, bypassing the toolbox functions and making the final trajectory smoother and more normal.

3) Trajectory Planning

In this part, we use MATLAB-robotics-toolbox to do the trajectory planning.

We used specific data for testing. After several experiments, we could get the curves of Angle, angular velocity and angular acceleration of each joint with time. It was found that the curves were smooth and had certain effects.

4) Control Procedure

The actuator we used is the AX-12A dynamixel connected with u2d2.

To control the actuator with MATLAB, we use matched dynamixeSDK package provided in the dynamixel official website. With the package we could simply control the actuator with the method $write2ByteTxRx$ which could write the goal position and moving speed in the actuator after initializing the baud rate and port number with your computer. To decrease the error and detect whether the actuator reach its goal position as written ,we use $read2ByteTxRx$ method to read the present pose to monitor. Since the two robot arm are in parallel, its necessary to cooperate the torque in both side or it wont rotate. To cooperate, we set complementary goal position in every two actuator. In practice ,since the camera catch the motion of people in a very high fps and the delay time due to read and write between VScode and MATLAB, we write the moving speed as a relatively slow number and pause after 4 seconds to avoid huge amount of tedious statistics. Also, the boundary is necessary in case the inverse kinematics calculated before gives a unreachable pose due to the restriction of actuator and mechanical configuration limitation.

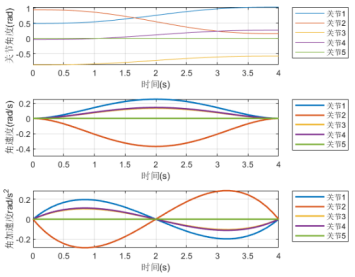

Fig. 17. Trajectory planning

Fig. 18. Control Process

5) Pressure Sensor

In this project, We successfully realized the remote control of the robot arm to move through visual recognition of human body posture. For real application, the one-way controlling is not enough, the manipulator needs to obtain the parameters perceived by the end-effector of the robot arm during operation, such as visual and tactile information. We try to add the tactile sensing module, so that we can get the tactile information of the end-effector when operating the object, so that the operator can get the tactile feedback, just like the real operation of an object in our daily life.

Due to the time constraints of this course project, we have placed the implementation of the tactile module last in our priority list. After completing the visual recognition part of attitude tracking with my teammembers, I began to study the force tactile sensing module of the manipulator end-effector.

The first step is to select the appropriate pressure sensing module and sensing mode. According to the specifications of the teaching kit we used and the working capacity of the robotic arm we assembled, we selected the pressure sensor measuring range of 20g – 6000g. This range also meets the pressure requirement of our daily grip on small objects. The sensor we use is the thin film pressure sensor on Taobao, whose response time is less than 10ms.

The second step is to detect the serial port information of the pressure sensor. In this project, the operation of the mechanical arm and the force tactile detection were carried out simultaneously, so I used Arduino as the independent detection of the pressure sensor in order not to interfere with the process.

For the convenience of reading data through serial port, we use the resistance-voltage conversion module of the thin-film pressure sensor, which can output analog signals, and adjust the magnification and comparison threshold to adjust the reading range. We use the AO pin (which is suitable for detecting changes in pressure trends and for measuring pressure values) to read the sensing to detect values, which range from 0.1V to 3.3V. In order to test the accuracy of the results, we use AO_res potentiometer to adjust the gain. In this step, we conducted a precision adjustment experiment by applying a known force and adjusting the potentiometer so that the serial port output weight is a given pressure. From the experimental measurement, we can get the relationship between the output voltage of the sensor module and the output resistance of the thin film pressure sensor:

Where U_0 is the output voltage of the sensor module, r_AO-res is the feedback resistance, and R_x is the output resistance of the thin film pressure sensor. I wrote a relatively simple program for every 10ms output voltage and pressure magnitude.

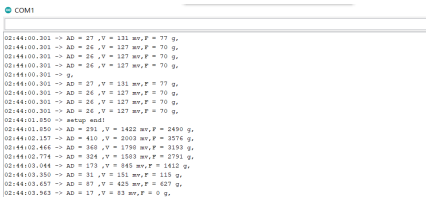

Fig. 19. Code of control

The third step is to apply the membrane pressure sensor to the end-effector grip force feedback. When the end-effector executes the command of grasping, we can obtain the force applied by the actuator to the object through the pressure sensor, so as to protect the object we grasp. For example, with brittle materials, too much force will cause the object to break.

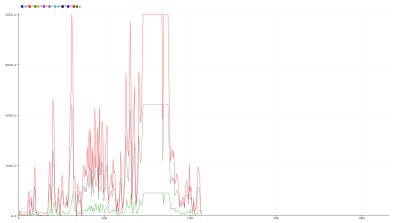

Fig. 20. Pressure curve

III. Experiment Results

1) Results

After numerous failures on everyone's part, we were able to integrate what we had and ultimately achieve much of what we had originally envisioned.

We calculated the visual control of the human arm in Python at the present, and exported the CSV file, which was the joint coordinate with equal interval. MATLAB read the real-time CSV file and then calculated the forward and inverse kinematics to get the rotation angle of each joint of the manipulator. In MATLAB by $while$ loop to control the motor, to each joint a target angle and angular velocity of motor, smooth mechanical arm's reach manpower needs constant motion position and the current position, and through the change of the hand movements to determine whether to the person in grasping object, and through the size of the pressure sensor in real time adjustment, so that the grasping object will not be damaged. The final experiment results are good, basically meet our needs.

Here are some of the renderings:

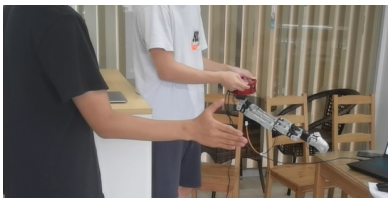

Fig. 21. Rendering1

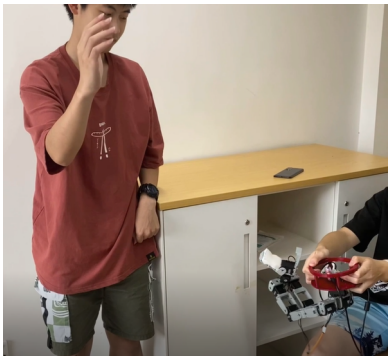

Fig. 22. Rendering2

2) Difficult and unresolved issues

Since this project is a course project, the time is relatively tight, and the time for us to try to implement the scheme is very short, so we need to spend more time to test the optimal scheme in some places.

One of the major problem is about robot arm control. We found in the test that when the human is not doing attitude control, the robot arm will have a certain Angle of deflection. This deflection is fine in the early stage, but after a certain period of operation, this effect will gradually accumulate, so that we can not distinguish whether the movement of the manipulator is controlled by our posture or its own influence. We have explored this problem to a certain extent and concluded three reasons for its influence. The first is the transmitted data. Since the results of visual recognition are all the world coordinates relative to the camera, when the human body keeps the same movement, the depth prediction will be affected by external interference factors and will change to a certain extent. Due to the size difference between the human arm and the mechanical

arm, the conversion coefficient calculated by us is only based on our theoretical calculation. So a small change in the depth of recognition is likely to result in a large movement of the arm. We found these movements by testing flat motion, such as bending the elbows and wrists. When we test the stereo position, such as the rotation of the shoulder and the rotation of the forearm, the deflection of the manipulator itself will increase. The second possible reason is the influence of the control mode of the manipulator. There are two ways to move our manipulator, one is constant speed movement, using angle control, the other is through angular speed control. In the process of operation, under the first constant speed control, the movement range of the manipulator will be very large, and there will be a certain degree of tremor, but the response speed will be very high. However, after the speed decreases, the motion stability of the manipulator will be improved. The second angular velocity control is more accurate and stable, but the response speed is slower than that of angle control. More qualitative experiments are needed to analyze this most suitable control parameter.

The third possible reason, similar to the first, is about the transmission of data. When the pose data is transmitted to Matlab for motion calculation and motor control, all the identified human posture information is transmitted, and the response speed of the manipulator is constrained by the hardware, so there is no way to respond quickly to all samples. This leads to the influence of the next action, especially the noise, when one action is not finished steadily in the angle control. We modify the manipulator by reducing the sampling frequency, at this time, the action of the manipulator will be more stable, but we have not found the most suitable method to filter the data, so we will filter out some important attitude information.

The second problem is about data transmission. Our human posture recognition is carried out using python, while the control is using matlab. Because of the incompatibility of the software, the way of real-time data transmission is very rough. Real-time reading of python exported data by Matlab will cause a delay in the control of the manipulator, which will behave badly when running 4min. We only use python in she xiang, but we haven't found an optimized motion solution for pytho. This problem needs to be solved by a better solution.

3) Future Expectation

At present, the mature control mode of mapping human arm to robotic arm is to use wearable equipment. The posture of the human arm is sensed by sensors, and then mapped to the robotic arm, such as the current industry-leading Haptex. Such a control method requires the operator to wear heavy sensing and control modules on the hand, although it can bring the operator the most fine tactile perception, but its weight and complicated operation make the comfort poor, and it is difficult to popularize and apply. Using visual recognition to obtain the attitude information of the operator can greatly reduce the complexity and cost of the equipment, especially when we can directly identify the depth information through the ordinary camera, we can popularize this operation in the largest range, so that ordinary users without professional equipment can also use this solution for remote control or virtual control of VR. At present, the attitude recognition data acquisition has been relatively perfect, and it is the part of the kinematics scheme that needs to be optimized. We will test different algorithms and find that the mapping relationship is most suitable for human and six-axis manipulators. We plan to rely on unity and steam VR for further hand posture modeling to achieve accuracy and stability.

APPENDIX A
DIVISION OF LABOR AND CONTRIBUTION

Member	Student ID	Contribution
Jia-jun Long	12011624	20 %
Yi-fei Chen	12010502	20 %
Yi-jun Fang	12011301	20 %
Zi-meng Wang	12011711	20 %
Ke-run Xiang	12012005	20 %

ACKNOWLEDGMENT

The authors would like to thank for the course ME331 which gave us time and equipment to do this project of visual control robotic arm and force tactile feedback of gripper. Also, the authors would like to thank the teacher and TA of this course, who gave us a lot of help. Again, thanks a lot!

REFERENCES

[1] V.N.Iliukhin,K.B.Mitkovskii,D.A.Bizyanova,A.Akopyan(2017), The Modeling of Inverse Kinematics for 5 DOF Manipulator,*Procedia Engineering*,176,498-505

[2] $website : https : //google.github.io/mediapipe/solutions/holistic\#python-solution-api$