Solving Rubik's Cube with Denso and DC motor

Yunchu Zhang, Jiahao Li, Boan Tao

***Abstract*— Vision robotic system plays an important role in manufacturing and industrial automation. This project aims to build such a system to solve complex Rubik's Cube problem with Denso and DC motor. By virtue of several image processing and control methods, the Rubik's Cube solver system works successfully.**

I. Introduction

In modern industry, robotic arms are increasingly being used to enhance effective productivity in dangerous or accuracy required work. Those robotic arms with integrated system have strong competitive advantage over "blind" ones. First, with the assistance of vision system, they are flexible in functionality. on the contrary "blind" robots can only execute a series of simple and repetitive task. They need reprogramming for different tasks in respect of location, size and function. Second, vision system ensures the safety of human-machine collaborative work by identifying nearby humans and turning off or slowing down to avoid collision. Vision robotic system has variety of applications such as assembly, packaging and palletizing which need integrated camera locating or reading objects and then guiding robotic arm's motion.

II. Objectives

- Accurate color detection of each grid of Rubik's Cube with Web camera.By means of image processing methods, image results should be immune of position and intensity of light source and environment noise.
- Effective trajectory planning and following of Denso robotic arm by applying forward and inverse kinematic equations and solid control method.
- Real-time grippers position and force control to grasp Rubiks Cube and cooperate with Denso robotic arm .
- Precise decentralized position control of DC motor as well as the bottom surface of mounted Rubiks Cube using designed PWM control signal input.
- Practical 3D printing models of Rubik's cube holder, gripper's rod and camera base plate.

III. Hardware system

In our system, DC motor was controlled by myrio and Denso robot arm ,gripper are controlled by laptop. Also, web camera acts as a set of eyes that detect color and position of a randomly shuffled Rubiks cube. To connect the robot arm and gripper, we plug in their cables to computer and distribute different IP address which could solve the conflict problem when sent control commend to them at the same time. As to the DC motor part, encoder is used to count the angle that motor rotate and sent those information to myrio. Both of them are build in one labview project and the codes are in different part(myrio codes at myrio folder, main.vi at my computer folder)

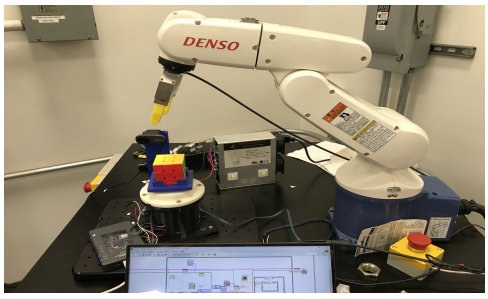

Fig. 1. Whole equipment

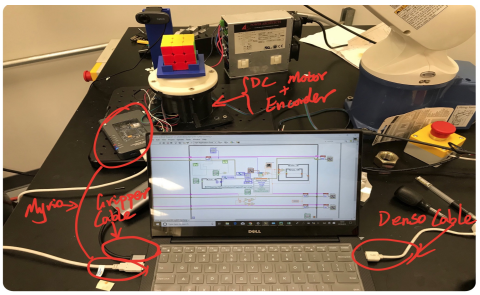

Fig. 2. Hardware

IV. Software system

To implement the system, Matlab, Labview are used for coding. Firstly, through web camera, rubik's cube's color could be detected and record in labview's memory. After that, through image process and a open source labview rubik's cube solver [1], the solution could be attained. Then, the solution will be sent to Matlab via serial communication. Then, in conrtol main loop, denso, gripper and motor will be controlled continuesly.

In matlab, we utilize inverse kinematic to get the certain rotation commands and transform those commands into the form that denso, gripper and motor required.

V. Method

A. Vision processing

Because web camera stores images in RGB plane, to reduce bit requirement and increase processing speed, chroma

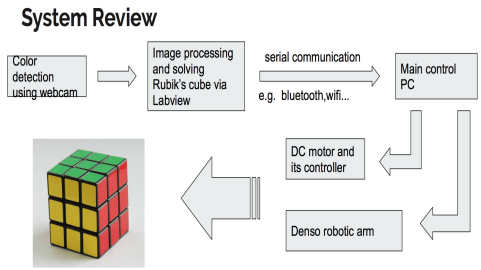

Fig. 3. Software

components are sub-sampled with respect to the black-and-white or luma component. In other word, the image obtained is converted from RGB to YCbCr via Labview program basing on following equations [2]

$$Y = \frac{77}{256}R + \frac{150}{256}G + \frac{29}{256}B \tag{1}$$

$$C_b = -\frac{44}{256}R - \frac{87}{256}G + \frac{131}{256}B + 128 \tag{2}$$

$$C_r = \frac{131}{256}R - \frac{110}{256}G - \frac{21}{256}B + 128 \tag{3}$$

To remove Salt and pepper noise generated by image sensor, transmission channel and reduce interrupt from environment noise, median filter toolbox in Labview is applied to process the image data. Then, area samples in 9 grids in each surface of Rubiks Cube are extract to compare with threshold value (brightness Y and color difference C_rC_b) for different color. Basing on those sample result, 6 colors are assign to each grids correctly. For different light source, threshold can be adjusted by average 6 color samples.

B. Kinematics

To solve a Rubik's cube in a random configuration, we need to control the Denso robotic arm to follow a designed trajectory. After solving for a solution by vision part of the system, we would break the whole solution down into several isolated part. The arm and the controller would execute each isolated part of the solution separately which is expressed in alphabet R and R' (Rotate right face (counter)clockwise), L and L'(Left), U and U'(Up), D and D'(Down), B and B'(Back), F and F'(Front).

The trajectory planning will be based on cartisian space trajectory, which is aimed to control the end-effector of the arm to move in a straight line between via points which is exactly what we want in the planning part while designing for each trajectory of the step.

The trajectory planning is based on kinematics of robotic arm. The forward kinematics problem is related between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. The joint variables are the angles between the links for revolute or prismatic joints, and the link extension in the prismatic or sliding joints. A systematic way of describing the geometry of a serial chain of links and joints was proposed by Denavit and Hartenberg and is known today as Denavit-Hartenberg(DH) notation. [3]The matrix A representing four movements is found by postmultiplying the four matrices giving four movements to reach frame j-1 to frame j in fig.4 Transformation between

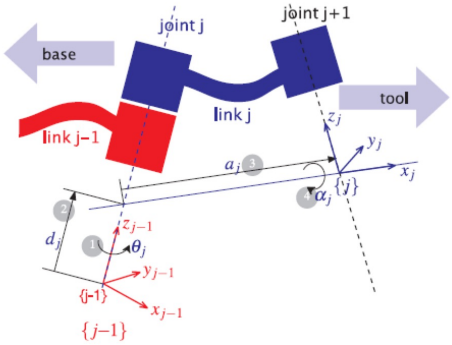

Fig. 4. DH representation of a general purpose joint-link combination

two joints in a generic form is given in

$$^{j-1}A_j = \begin{bmatrix} cos\theta_j & -sin\theta_j cos\alpha_j & sin\theta_j sin\alpha_j & a_j cos\theta_j \\ sin\theta_j & cos\theta_j cos\alpha_j & -cos\theta_j sin\alpha_j & a_j sin\theta_j \\ 0 & sin\alpha_j & cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Denso Robot is a 6 degree-of-freedom robotic manipulator. The link lengths are given in Figure.5. World frame and joint frame used in calculations are shown in Figure.6

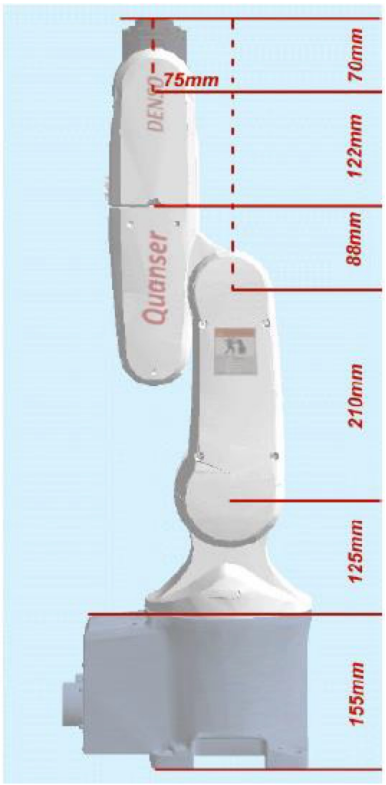

Fig. 5. Link Length of Denso Robot

The following table shows DH parameters of the Denso robot arm necessary to derive the kinematics of the robot.

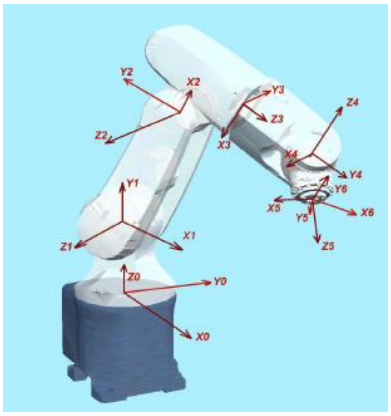

Fig. 6. World frame and joint frames

Gripper is not included in the analysis.

i	θ_i	d_i	a_{i-1}	α_{i-1}	Joint Limits
1	q_1	d_1	0	90	-160,160
2	q_2	0	a_2	0	-120,120
3	q_3	0	a_3	-90	20,135
4	q_4	d_4	0	90	-160,160
5	q_5	0	0	-90	-120,120
6	q_6	d_6	0	0	-360,360

where $d_1 = 0.125m$, $a_2 = 0.21$, $a_3 = -0.075$, $d_4 = 0.21$ and $d_6 = 0.07$, and the length of our gripper is 0.12m.
The homogeneous transformation matrix from base frame to end-end-effector could be expressed as

$$ {}^0_6T = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

We will use the robotics toolbox in MATLAB to calculate the homogeneous transformation matrix. For the inverse kinematics of the arm, also use the *SerialLink.ikine* function in the MATLAB Robotics Toolbox. For more details, see the code attached in the appendix.

C. Control

In control part, the main idea is trajectory following and pose maintenance of the center the Rubiks cube by decentralized position control of Denso robotic arm using inverse dynamics. Since the Denso arm's package has contained the controller, we only need to input trajectory pareameters and put the face to be rotated in the bottom for each step. Besides, we design a decentralized position controller (from lecture)to control the mount of Rubiks cube to conduct rotation of each step of solution. Finally, We will use Grippers force sensor to control force quantity when grasping object and avoid rubik's cube slipping.

The controller are designed as figure 8 shows. The motor will set zero position until it receives rotation command. And error between the set command and encorder number could be transformed into PID controller. Then the PID controller will output duty cycle to motor. Before the motor receive rotation command,denso will send rubik's cube to the above

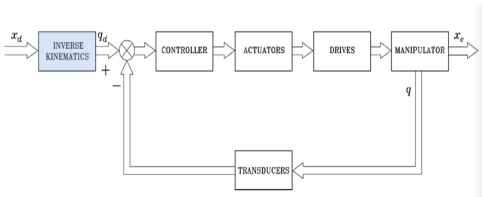

Fig. 7. Lecture Note for decentralized control

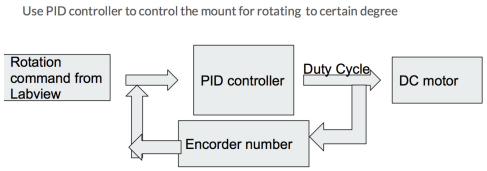

Fig. 8. PID controller

of the mount.Then, the gripper will release and then grape it again to make sure rubik's cube is in the certain position of mount.After these series of action, motor will rotate it to right position. As you can see in the figure bellow, there exists interspace between Rubik's cube and mount since the denso robot arm has some little error when it be controlled to one position. These interspace will result the steady error in rotate action. Thus, we tuned the PID parameter especially the Integration part to figure out this problem.

Firstly, we set a large P parameter 0.1 from the figure, it has a big vibration.

Then, we set a small P parameter 0.009 from the figure, there is no vibration but has a big steady error, which means that the P is not big enough to decrease the error. And then we try many times to get a good P=0.005, in this case, the plot not only does not have vibration but has little steady error and quick response. In the next step, we add PD control (P=0.005,D=0.00011)to get quick response and little overshot. At last, to overcome overshot and steady error in hardware's limitation, we add I=0.005 to accumulate error

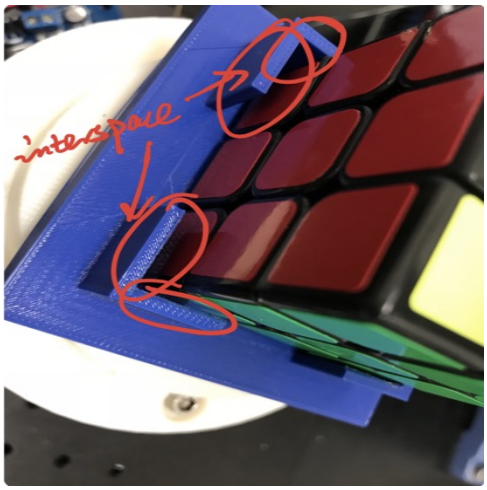

Fig. 9. Interspace between Rubik's cube and mount

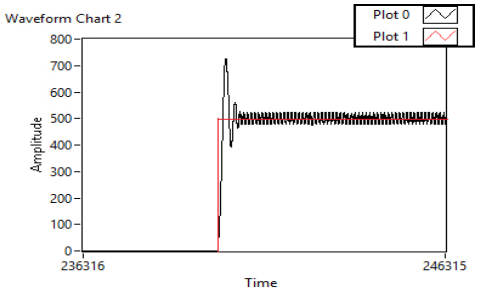

Fig. 10. Large P

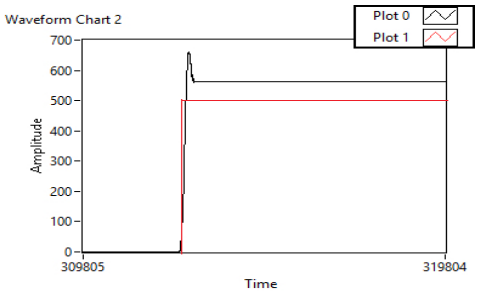

Fig. 11. Small P

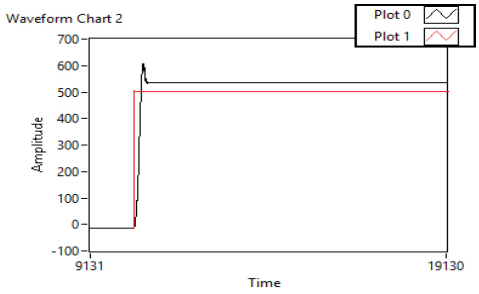

Fig. 13. Good PD

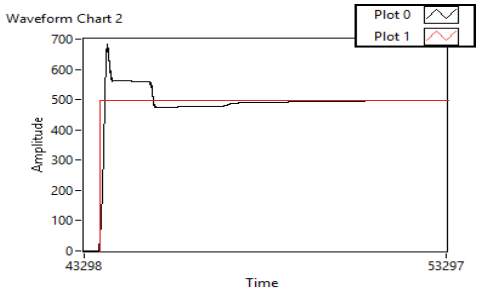

Fig. 14. Good PID

and could have response when error is very small(such as 1 2 degree).

Since myrio and laptop's code could not be run at the same time, thus, a zero button need to be put in the whole project. To achieve real-time control, I design a special logic like the figure bellow.

Firstly, the myrio code runs and set zero position for the motor. Then, the control part runs and make initialization. At last, when I click one button, the control loop could sent commend to 3 end-effector together. In control's main loop, we set 2000 ms time delay to make three end-effectors could receive same sequence's command in one control loop.

VI. Results

A. Color Detection

By average samples of 6 colors, thresholds range of them are determined shown below. Corresponding color detection result reveals that the whole program works precisely and responsively. The colors of cubes in each surface are then stored in the form of array which are sent to solver Labview program.

B. Trajectory Planning

By using the MATLAB Robotics Toolbox, we could get how the joint angle of each joint changes while executing different steps. The following shows the time history of each joint angle for each solution.

The reason why the joint angles do not change in D is that while executing this step, the robot arm will not move. Instead, the DC motor will rotate the face.

The results show a good performance of trajectory planning using Cartesian space method for the robot arm while executing each step of solution, it could successfully avoid collision with the environment in the implementation.

C. Motion Control

From the output about PWM in motor, we can see that, it shows the important function when the Rubik's cube was put or moved from the mount.It could make mount at stable

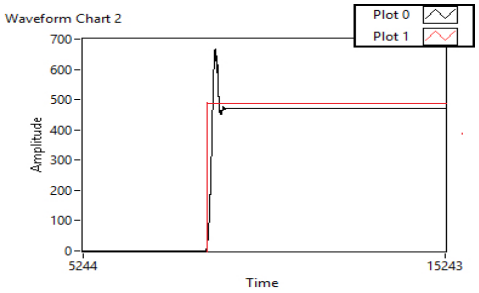

Fig. 12. Good P

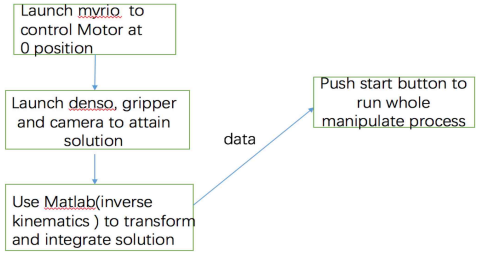

Fig. 15. Real time process

		Y	Cr	Cb
Blue	max	109.7	114.5	190.8
	min	057.5	085.0	149.5
Red	max	070.2	192.7	125.1
	min	042.9	148.1	099.6
Orange	max	133.2	199.5	106.1
	min	082.9	156.9	078.9
Yellow	max	205.4	161.8	094.9
	min	118.1	116.4	053.3
Green	max	142.0	124.0	130.5
	min	097.7	091.2	079.8
White	max	200.3	142.1	141.9
	min	134.1	115.4	115.1

Fig. 16. Threshold value of 6 colors

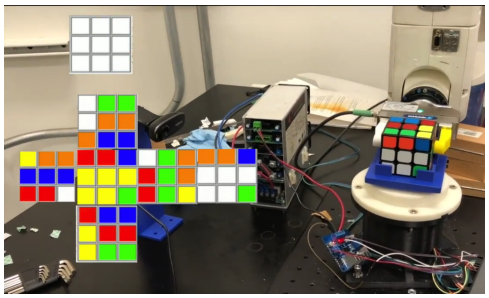

Fig. 17. Color detection result

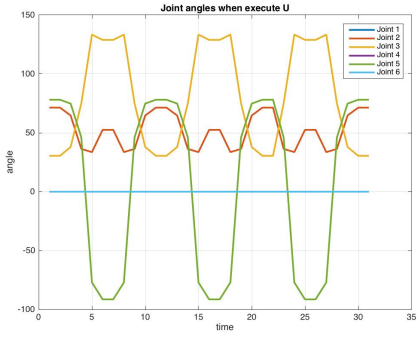

Fig. 18. Joint angles of U

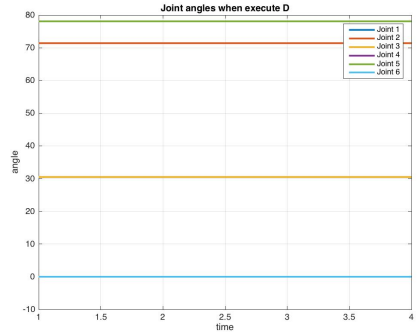

Fig. 19. Joint angles of D

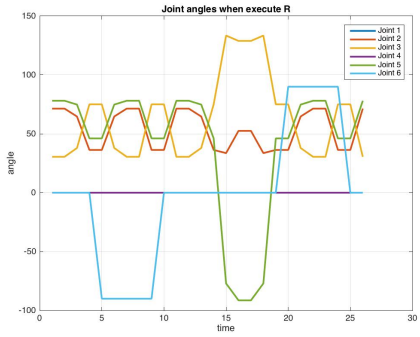

Fig. 20. Joint angles of R

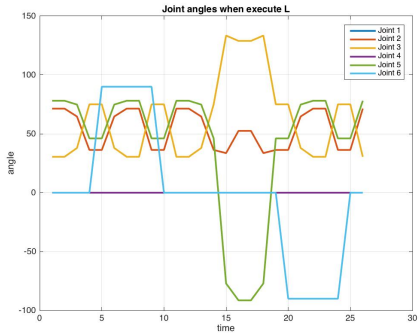

Fig. 21. Joint angles of L

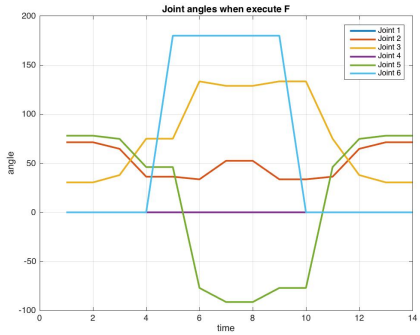

Fig. 22. Joint angles of F

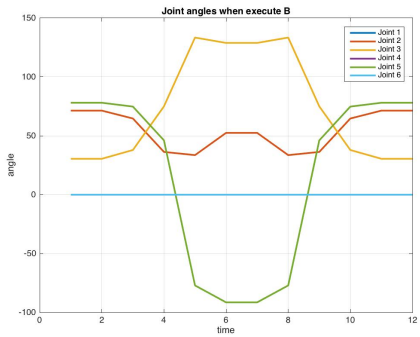

Fig. 23. Joint angles of B

position and resist being moved. The little vibration means the resist process.

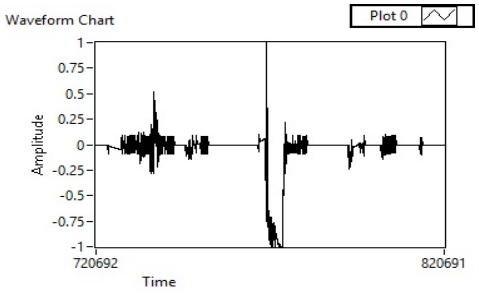

Fig. 24. Result in PWM output1 which could indirectly reflect error time history

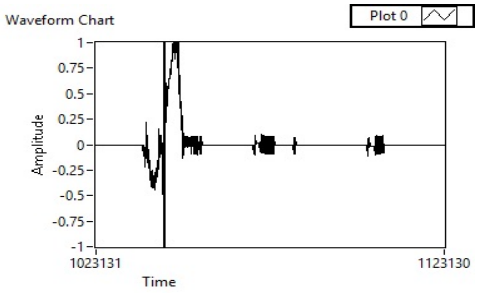

Fig. 25. Result in PWM output2 which could indirectly reflect error time history

The whole result will be shown in our video demo. Having seen that demo, you could conclude that our controller could achieve the basic rotation goal in a quick response.

VII. Discussion

A. Capabilities

- Precise color detection of each grids of Rubik's Cube and acquisition of corresponding solutions via open-source solver Labview program.
- Effective trajectory planning with forward and inverse kinematic equations to Denso Robot arm.
- Real-time control and smooth coordination of Denso robotic arm, gripper and DC motor.
- Direct-Force control and decentralized position control with PID.

B. limitations

- Manual change of threshold value of colors basing on different light source.
- A gap between real error and compute error in Rubik's Cube's position caused by inter space between mounting base and Rubik's Cube
- Limitations on control the Denso Robot arm since it has own package which could not be changed.

C. Future improvement

The design can be further improved in following respects. First, the accuracy of hardware equipments need to be improved to ensure more accurate position control. For example, the square solid mounting base is replaced by a kind of hollow four-sided clamp. Second, a model-based controller such as state-feedback observer control is designed so that logical analysis of the whole system can be assessed. Third, two Denso robotic arm are applied instead of motor to complete the task as a simulation of industry working space. Fourth, position of edges are detected via web camera to check whether each step is perfectly complete. Then position data as feedback is sent into the controller design to optimize system's performance.

VIII. Conclusion

As the Denso robotic arm, gripper, DC motor and 3D printing service are provided by laboratory, total cost of the project is around 100 dollars for web camera and other tools like tapes and lubricating oil. The whole robotic system achieves the goal successfully which can also extend to other applications. With the assistance of vision system feedback, accurate trajectory planning and precise position control methods, it meets most of requirements of industrial utility.

IX. Appendix

As to Denso arm and gripper, we strongly recommend that there should be a manual for guiding students to utilize it quickly. Since it took us too much time on figuring out connection and communication methods among those hardware equipments and computer. Also, another thing need to do is trying to get access to source code of Denso package library. With that code, more control methods can be implemented basing on it.

A. Contribution

All members take part in design and tune PID controller.

- Yunchu Zhang: Whole hardware connection and communication. Whole Software integration and LabVIEW coding to implement following tasks: a) Use LabVIEW to control the Schunk Gripper (e.g. given an input (defined whatever you like) to control the gripper to grip the rubik, such as making the distance b/n the two finger 56mm. b) Given joint angles, use LabVIEW VI to control Denso robot arm. c) Control DC motor with decentralized position control method. d) Collaborate with Boan to figure out how to combine the computer vision part and the control of denso part.
- Boan Tao: Mainly design vision system. a) Hardware selection and purchase. b) Communication between environment and computer via web camera. c) Real-time image processing and color detection using Labview. d) Solve the Rubik's Cube and send corresponding solution to main control computer by series communication basing on motion criteria formulated by Jiahao.
- Jiahao Li: Design a base for the camera that Boan chose. Decide the best position for the rubik to reach a best

manipulability. Design the trajectory for each action of the Denso a) Rotate the upper surface in clockwise direction for 90 degree. b) Flip the rubik in order to rotate the right surface Do a simulation in MATLAB for each action.

B. Labview and Matlab code Screenshot

Following figures are screen shot of some parts of Labview and Matlab program and corresponding code. The detail of them can be looked up in the submitted folder.

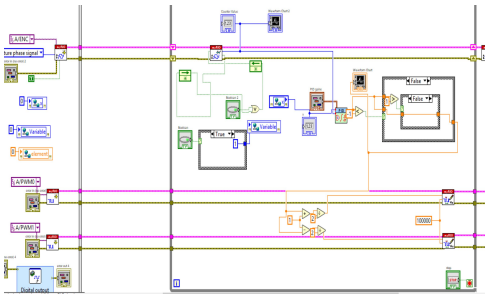

Fig. 26. Motion control Labview code

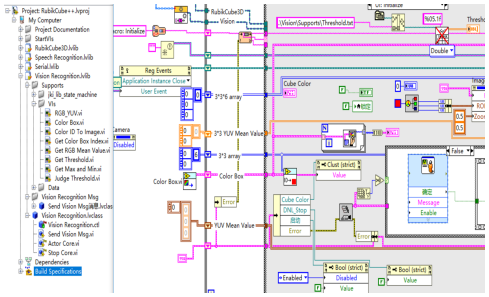

Fig. 27. Vision system Labview code

```
clear;
%            theta d a alpha
denso(1) = Link([0  280 0    0       ],'modified');
denso(2) = Link([0   0   0   -pi/2   ],'modified');
denso(3) = Link([0   0  -210 0       ],'modified');
denso(4) = Link([0  210 -75  pi/2    ],'modified');
denso(5) = Link([0   0   0   -pi/2   ],'modified');
denso(6) = Link([0  70   0    pi/2   ],'modified');

% L_T = 120; % true
L_T = 120;
tool=transl(0,0,L_T);
densoMod = SerialLink(denso,'tool',tool);

% densos(1) = Link([0  280 0    pi/2],'standard');
% densos(2) = Link([0   0   0    0],'standard');
% densos(3) = Link([0   0  -210 -pi/2],'standard');
% densos(4) = Link([0  210 -75  pi/2],'standard');
% densos(5) = Link([0   0   0    -pi/2],'standard');
% densos(6) = Link([0  70   0    0],'standard');
% densosMod = SerialLink(densos,'tool',tool);
% fKineTransMatr = densomod.A([1 2 3 4 5 6 7],[t1 t2 t3 t4 t5 t6 0]);
% q = [pi/6 -pi/2 pi/15 pi/3 pi/6 pi/6 0];
qq = [0 0 0 0 0 0];

q0 = [-0.0000    2.9407   -0.7455   -0.0000    0.9464    0.0000];

% the initial value of q to ensure a better configuration
```

Fig. 28. Part of Matlab code

REFERENCES

[1] Brownan,Rubiks-Cube-Solver,2017,GitHub repository,https://github.com/brownan/Rubiks-Cube-Solver

[2] Charles Poynton, Digital Video and HDTV, Chapter 24, pp. 291292, Morgan Kaufmann, 2003.

[3] Mehmet Erkan and Memik Taylan, Forward and Inverse Kinematics Analysis of Denso Robot.