



Report of Lab Section

Version 1.0 – 11/27/2017

Tingyuan LIANG, Guang CHEN

This document is the report of ELEC5640 lab section, a design called Colobot. Colobot is designed based on Stäubli TX60, integrating computer vision and robotic control. The information contained in this document is tentative, and is provided solely for planning purposes of the recipient. The features described for this design released are likely to target at basic research and there is still large improvement space. Our group assumes no liability or responsibility for decisions made by third parties based upon the contents of this document.

Table of Contents

Section I. Introduction of Lab Section	3
Basic Introduction of Stäubli TX60	3
Content of Lab Section	3
Section II. Robot System Design	4
Subsystems in the Design	4
Obtainment of Components in the Design	5
Section III. Detailed Implementation of Subsystems	6
Mechanism: Design of Gripper and 3D Printing	6
Mechanism: Fundamental and Interconnection of Pneumatic System	7
Mechanism: Motion of Robot Arm	8
Position of the End-Effector	8
Orientation of the End-Effector	9
Low-Level Control: Serial Port Control of Solenoid Valve	11
Master Computer Side	12
Slave Computer Side	13
Computer Vision System: Setting Up	14
Calibration of the Wide-Angle Camera	14
Configuration of Project for OpenCV	15
Computer Vision System: Recognition of Target Cube	16
Characteristics of Target Cube	16
Basic Recognition Algorithm: A Color-Based Detection Algorithm	16
Optimized Recognition Algorithm: Hybrid Detection Algorithm	18
Computer Vision System: Recognition of Box	19
Section IV. Strategies to Finish the Task	21
Step I: Search for the Box	22
Step II: Search for the Cube	22
Step III: Pick Up the Cube	24
Step IV: Move around the Obstacle	24
Step V: Search for another Cube	25
Step VI: Place the Cube	25
Acknowledge	25

The Part Done by Tingyuan LIANG .

The Part Done by Guang CHEN .

The Part Done in the Cooperation .

Section I. Introduction of Lab Section

Basic Introduction of Stäubli TX60

In the sixth week of ELEC 5640, the section lab focuses on some basic operation skills of Stäubli TX60, demonstrated in **Figure 1**.



Figure 1 TX60 6-axis industrial robots

Stäubli is a mechatronics solutions provider and the TX60 series industrial robots feature an articulated arm with 6 degrees of freedom for maximum flexibility. Detailed information of can be found in the line shown below:

<https://www.staubli.com/en/robotics/6-axis-scara-industrial-robot/low-payload-6-axis-scara-robot/6-axis-industrial-robot-tx60/>

Content of Lab Section

The design in the lab section should meet the requirements which can be divided two parts as shown below, which are related close to each other to a large extent:

- The system should use the camera to get the position and orientation of the target object and the obstacles.
- The gripper has the ability to manipulate the target objects, the pick-and-place procedure is operated freely.

In the lab section, the target object is a red cube, size of which is $6\text{cm} \times 6\text{cm} \times 6\text{cm}$, and the obstacle is a box ($80\text{cm} \times 80\text{cm} \times 80\text{cm}$). To make it clear, both of them are marked in **Figure 2**.

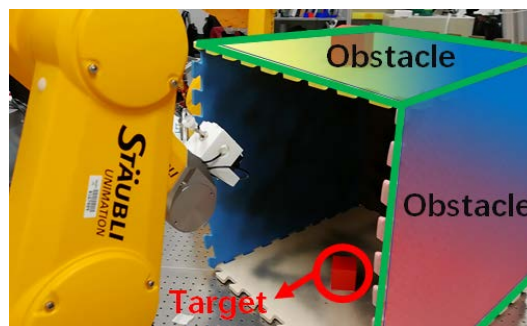


Figure 2 The target cube and the obstacle box in the lab section

To crystallize the procedure of the task, we list the steps as below:

- Search for the position the box, where the red cube is located.
- Search for the position the red cube
- Pick up the red cube
- Move around the box, the obstacle, to the roof of the box
- Search for the position another red cube on the roof of box
- Place the red cube on the cube positioned by previous step

Such steps are present in the flowchart, shown as **Figure 3**.

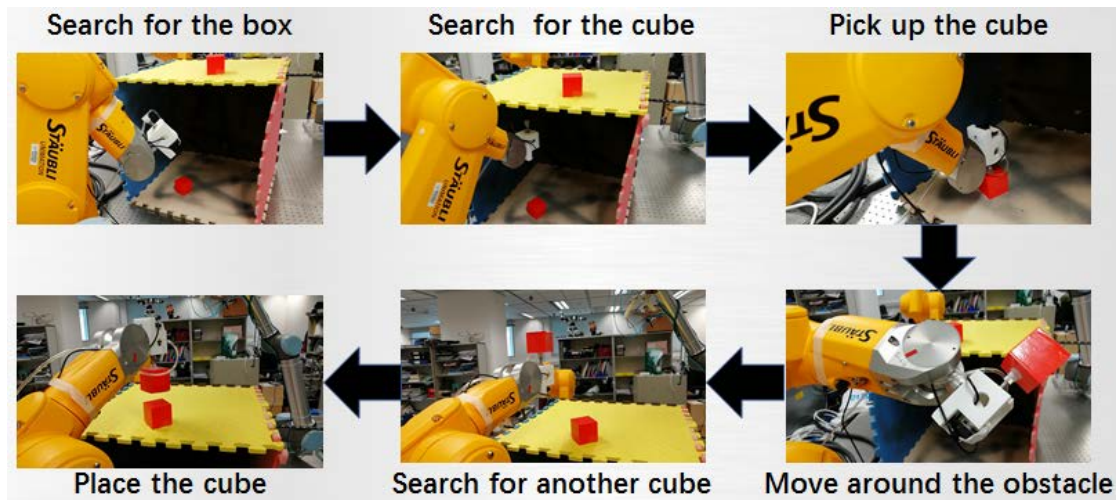


Figure 3 Execution flow of the lab task

Section II. Robot System Design

Subsystems in the Design

In order to meet the requirements of the pick-and-place task, a robot system, integrating computer vision and robot motion manipulation, has been designed. The system consists of six subsystems, including master computer system, slave computer system, robot arm, vision system and pneumatic system. Their main functions are listed in Table 1.

Subsystem	Components	Functions
master computer system	Acer Laptop	strategy, CV algorithm and robot arm control
slave computer system	Arduino Mega 2560	pneumatic control
robot arm	Stäubli TX60	motion
vision system	wide angle 720p camera, Laptop	image capturing and processing
gripper system	gripper, pump, vacuum generator, valve, suction pad, tube	pick and place

Table 1 Subsystems in the overall design

The fulfilment of the task relies on the interactions among the subsystems. **Figure 4** shows the control hierarchy of the system and the interactions in the system.

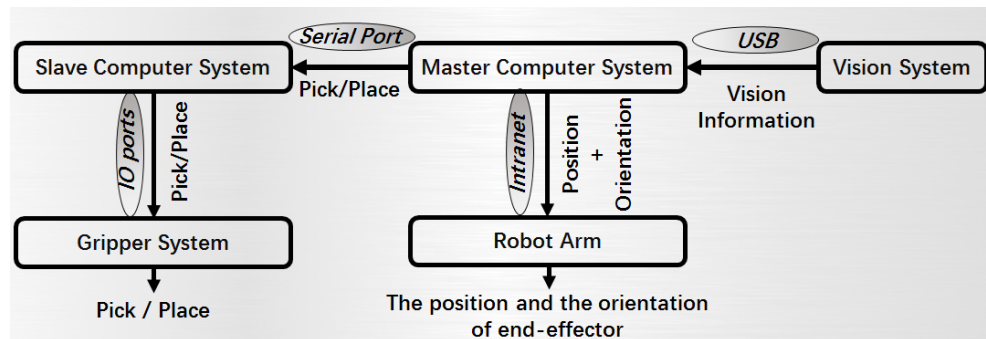


Figure 4 the control hierarchy of the system and the interactions in the system

As shown in the **Figure 4**, the master computer system is at the top of the system and the overall work flow is under the control of it. It obtains necessary from camera, positions particular targets and send commands to other subsystems. The slave computer system acts as an interface for master computer system to control the pneumatic system, sending the commands PICK and PLACE to gripper system via I/O ports. The robot arm accounts for the motion of the gripper so that the gripper can reach the particular point in the space according to the commands from the master computer system. Vision system is actually the camera, the most important sensor in the system, capturing the image and guiding the motion of robot arm. Gripper system is the exact executor of the commands PICK and PLACE.

Obtainment of Components in the Design

The laptop, the Arduino board, the camera, the air pump and the Stäubli TX60 are pre-obtained. The gripper structure is 3D-printed and the detail procedure of designing and printing will be illustrated in **Section III**.

Others components, including the pneumatic components, are obtained from online shopping. These components and related links are list in **Table 2**.

Item	Link
Suction pad	https://item.taobao.com/item.htm?spm=a1z09.2.0.0.4e387766nY3e2q&id=522137237286&_u=r3ms5219b46
Multi-Way Connector	https://item.taobao.com/item.htm?spm=a1z09.2.0.0.4e387766nY3e2q&id=553931356484&_u=r3ms52162f9
Vacuum Generator, Valve and Tube	https://item.taobao.com/item.htm?spm=a1z09.2.0.0.4e387766nY3e2q&id=530996880832&_u=r3ms521825b
One-Way Valve	https://item.taobao.com/item.htm?spm=a1z09.2.0.0.4e387766nY3e2q&id=39985966617&_u=r3ms521cc5d
Optical coupler	https://detail.tmall.com/item.htm?id=40885062911&spm=a1z09.2.0.0.6996ec05XLeypE&_u=s3ms5211b94&skuId=3131615196124

Table 2 Components obtained online and corresponding links

Section III. Detailed Implementation of Subsystems

In this section, the implementation of each subsystem, except robot arm, will be illustrated so that reader can re-implement the design flow of our robot system. The following content can be categorized into three parts, mechanisms, low-level control and computer vision.

Mechanism: Design of Gripper and 3D Printing

To attach the suction pad and the camera to the end-effector of the robot arm, we design the gripper shown in **Figure 5**, with Solidworks 2017. The motivation of the design will be mentioned later with concrete figures.

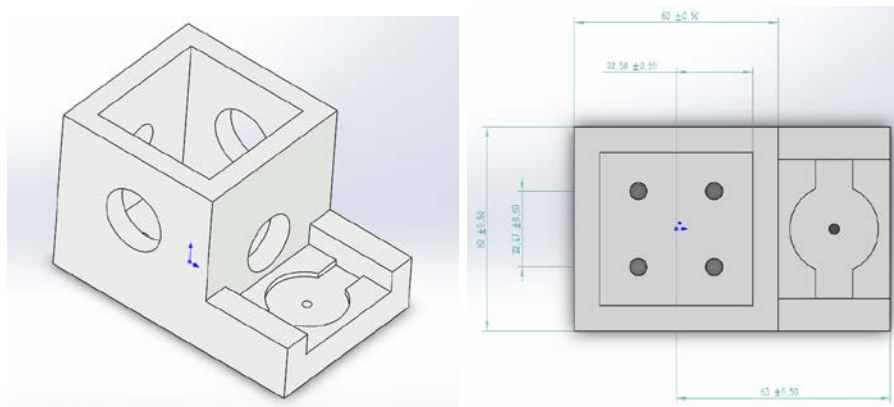


Figure 5 3D model of gripper in Solidworks 2017

To print the 3D model with 3D printer, steps, shown in **Table 3**, are required to be done in order on several different platforms.

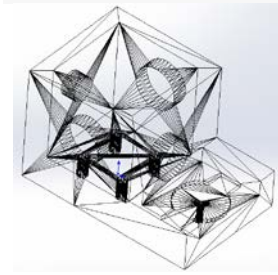
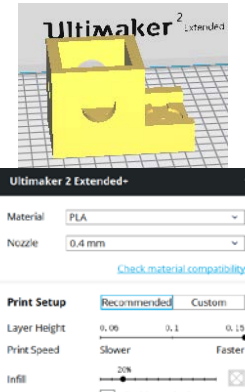
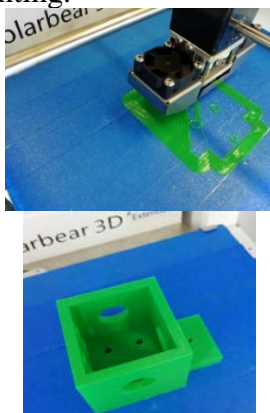
1. Solidworks 2017	2. Cura	3. Polarbear Printer
<p>Save the model as STL file</p> <p>保存到文件(s)</p> <p>将当前文件保存为 3D 打印常用格式。</p> <p>格</p> <p>STL (*.stl)</p> <p>保存文件...</p> 	<p>Import the STL file to Cura and set the configuration of printing. Save the gcode file to SD card.</p> 	<p>Plug the SD card into printer and start the printing.</p> 

Table 3 Steps to print a 3D model

After finishing the printing task, a 3D model will be obtained. The camera and the suction pad are located on the opposite surfaces of gripper, as shown as **Figure 6**.

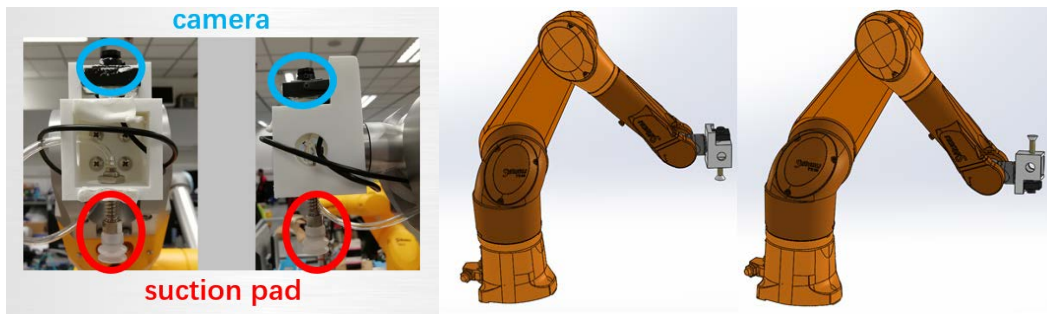


Figure 6 Motivation of the design of gripper

According to **Figure 6**, the motivation of our design of the gripper can be easily understood. When the camera orients downward and the target is located in the center of the image captured by camera, the target is under the camera. Then the end-effector of the robot arm will be rotated 180 degrees about its own axis and the suction pad will orient downward. Since the orientations of the camera and the suction pad are exactly opposite and the lens of camera is concentric with the suction pad, the suction pad will aim at the target after rotation. This design will reduce a lot of effort taken to handle the transformation matrix between the camera and the suction pad.

Mechanism: Fundamental and Interconnection of Pneumatic System

In the previous part of mechanism, the suction pad has been attached to the gripper. However, only a suction pad cannot pick up the cube and it has to be equipped with an appropriate pneumatic system so that it can get the abilities to pick and place.

Demonstrated in **Figure 7**, The pneumatic system consists of suction pad, vacuum generator, solenoid valve, one-way valve, air tube and air pump. The size of the suction pad needs to be calculated to make sure it is capable of picking up the cube. In our design, the diameter of the pad is 20mm and with a 3 Bar air pump it can pick up a 50g cube.

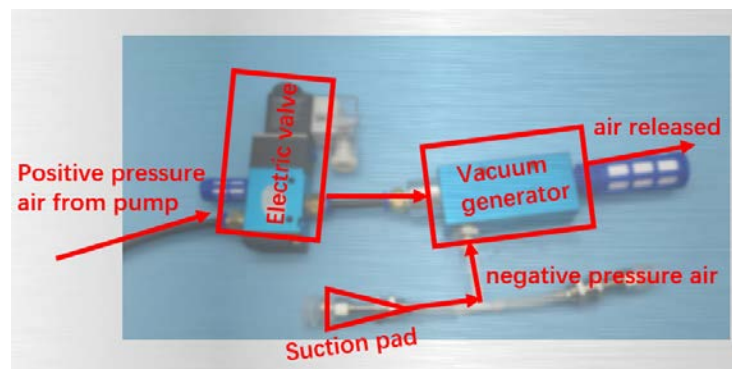


Figure 7 Pneumatic System

The working principle of this pneumatic system can be illustrated as follow: When the solenoid valve switches on, air with high positive pressure will be injected into the vacuum generator. According to Bernoulli principle, one of the ports of the vacuum generator will generate negative pressure so that the suction pad can pick the cube. In contrast, when the solenoid valve switches off, the negative pressure of the suction pad will be dissolved and the cube will be released from the pad. **Figure 8** shows how the pneumatic system implements the PICK action and the PLACE action.

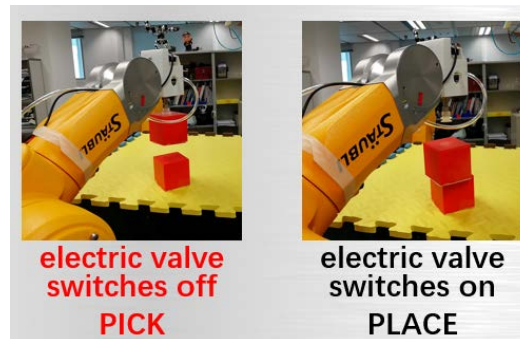


Figure 8 how PICK action and the PLACE action are implemented

Mechanism: Motion of Robot Arm

The motion of robot arm is implemented to maintain appropriate orientation and position of the end-effector.

Position of the End-Effector

The motion of changing the XYZ position of the end-effector can be implemented with a function, called *MoveCartesian*, in the library provided by Staubli. The input of the function is a vector containing six elements. The first three element are the XYZ coordinate offset of the end-effector relative to the origin of the Staubli TX60 body frame. The other three elements are actually a set of Euler angles which control the orientation of the end-effector. A typical application of the function *MoveCartesian* is shown in **Table 4**.

```
bool Move2XYZ(TX6OL* TX6OLObj, double * XYZ, double epsilon)
{
    std::vector<double> targetPos;
    targetPos.clear();
    targetPos.push_back(safe_rangeX(XYZ[0]) / 1000);
    targetPos.push_back(safe_rangeY(XYZ[1]) / 1000);
    targetPos.push_back(safe_rangeZ(XYZ[2]) / 1000);
    targetPos.push_back(DEGREE_2_RADIAN(0));
    targetPos.push_back(DEGREE_2_RADIAN(-90));
    targetPos.push_back(DEGREE_2_RADIAN(180));
    if (!TX6OLObj->MoveCartesian(targetPos)) return false;
    return IsArmReach(TX6OLObj, epsilon, targetPos, Cartesian);
}
```

Table 4 A typical application of the function *MoveCartesian*

As shown in Table 4, TX60L is a class containing a set of low-level API provided by Stäubli so that programmer can control the motion and check the status of the robot arm easily. These APIs, both source code and interface, can be found in the GitHub link shown below:

https://github.com/CRLab/staubli_tx60_controller/tree/711c3b9e3c70d45435affdbb03ecbe30ca298472/staubli_tx60_api

By setting appropriate values to the first three elements of the function input, a inverse kinematic will be calculated by the library and the rotation of each joint of Stäubli TX60 will take action to lead the end-effector to reach the target position. Such procedure has been already implemented by the low-level processes of Stäubli TX60 and we directly adopt their solution for the consideration of the safety of the Stäubli TX60 platform safety and the stability of control.

However, it is important to set a safe working space of position. Not only an external boundary surface need to be set so that the robot arm will not harm the items out of this boundary, but also an inner boundary surface is necessary to avoid that the end-effector hit the body of robot arm, damaging the structure of the gripper. A tragic example can be found in **Figure 9**, demonstrating the necessity of inner boundary surface. The function *safe_range* shown in **Table 4** accounts for the boundary check.

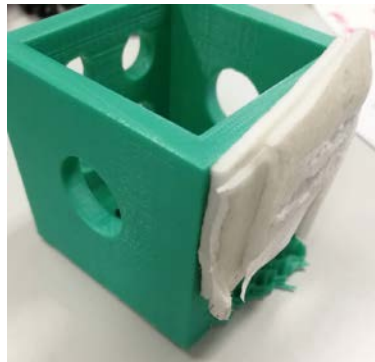


Figure 9 A broken gripper due to the absence of inner boundary surface

Orientation of the End-Effector

We also analyze the related solution to handle the orientation of the end-effector. Initially, the orientation of the end-effector is the same as the one of the base of the robot arm, as shown in **Figure 10**. Since we want that the camera and the suction pad can aim downward according to commands, the Z-axis of the orientation of the end-effector should be lay in the level surface, as shown **Figure 11**.

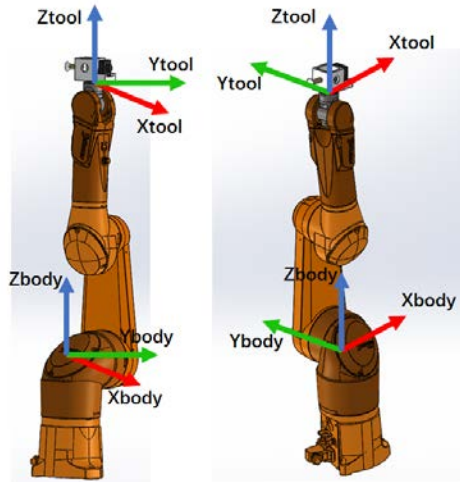
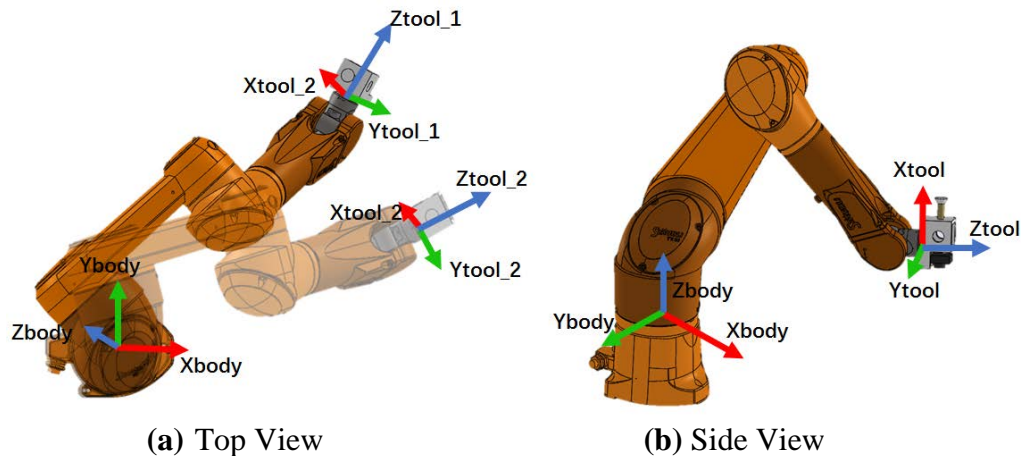


Figure 10 Initial Orientation of End-Effector



(a) Top View

(b) Side View

Figure 11 Target Orientation of End-Effector

The rotation of orientation can be represented by Euler angles. As for Stäubli TX60, the rotation of the orientation of the end-effector can be implemented via Intrinsic rotations. Intrinsic rotations are elemental rotations that occur about the axes of a coordinate system XYZ attached to a moving body. Euler angles can be defined by intrinsic rotations. The rotated frame XYZ may be imagined to be initially aligned with xyz , before undergoing the three elemental rotations represented by Euler angles. Its successive orientations may be denoted as follows:

- x - y - z , or x_0 - y_0 - z_0 (initial)
- x' - y' - z' , or x_1 - y_1 - z_1 (after first rotation)
- x'' - y'' - z'' , or x_2 - y_2 - z_2 (after second rotation)
- X - Y - Z , or x_3 - y_3 - z_3 (final)

For the above-listed sequence of rotations, the line of node N can be simply defined as the orientation of X after the first elemental rotation. Hence, N can be simply

denoted x' . Moreover, since the third elemental rotation occurs about Z, it does not change the orientation of Z. Hence Z coincides with z'' . This allows us to simplify the definition of the Euler angles as follows:

- α represents a rotation around the x axis,
- β represents a rotation around the y' axis,
- γ represents a rotation around the z'' axis.

To reach the target orientation in **Figure 11**, we have calculated the values of α , β and γ .

$$\text{For the situation where the camera aims downward, } \begin{cases} \alpha = -\frac{\pi}{2} \\ \beta = \text{atan2}(x, y) \\ \gamma = -\frac{\pi}{2} \end{cases} \dots (1)$$

$$\text{For the situation where the suction pad aims downward, } \begin{cases} \alpha = -\frac{\pi}{2} \\ \beta = \text{atan2}(x, y) \\ \gamma = \frac{\pi}{2} \end{cases} \dots (2)$$

The value of α is set to be $-\frac{\pi}{2}$ so that the Z-axis of the orientation of the end-effector can be lay in the level surface. The value of γ is set to be $-\frac{\pi}{2}$ or $\frac{\pi}{2}$ so that whether the camera or the suction pad aims downward can be controlled. According to the top view shown in **Figure 11(a)**, the value of β is set to be $\text{atan2}(x, y)$ so that the Z axis of the end-effector intersects with the Z axis of the Stäubli TX60 body frame. In this way, the direction of Z axis of the end-effector will be change in the level surface according to the position and keep aiming outward about the origin of the body frame.

Low-Level Control: Serial Port Control of Solenoid Valve

As mentioned previously, since the master computer system cannot generate IO signal toward pneumatic system directly, an Arduino Mega 2560 board has been applied in the system as an hardware interface of the pneumatic system. When the master computer system wants to pick up or place down the cube, it will send the corresponding command to the Arduino board. After decoding the command, the Arduino board will switch on/off particular I/O port connected to the optical coupler. As result, corresponding solenoid valve will be switched on/off and the suction pad can pick or place. Such procedure is depicted concretely in **Figure 12** and the actual interconnection is shown in **Figure 13**.

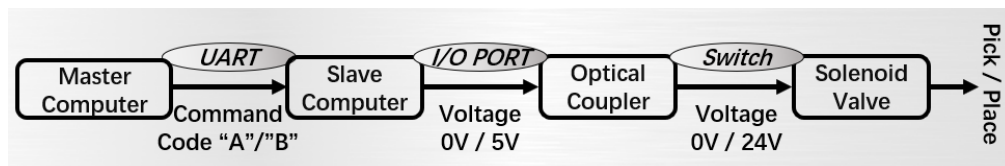


Figure 12 The flow of serial port control of solenoid valve

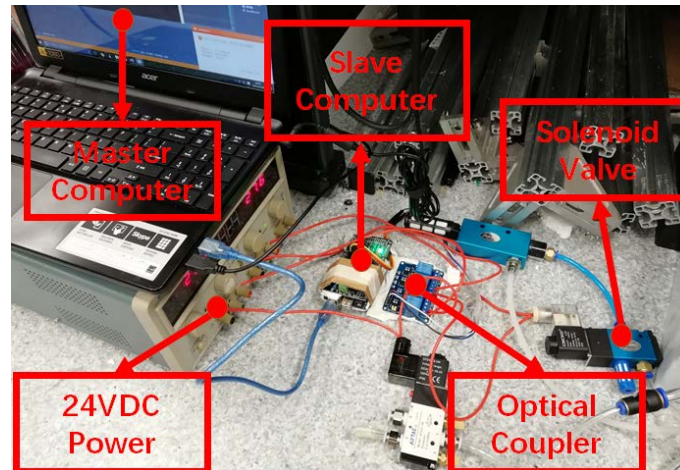


Figure 13 The actual interconnection of serial port control of solenoid valve

Master Computer Side

In detail, in the side of master computer system, the serial port is regarded as a file. For C++ in Windows, using the function *CreadFile*, we can open the USB UART, a I/O device handling the serial port communication between master computer and slave computer. Based on the handle generated by *CreadFile*, a function called *WriteFile* can send data to the USB UART. As shown in **Table 5**, If the master computer wants to pick up the cube, ten characters “B” will be sent to the slave computer. In contrast, to place down the cube, ten characters “A” will be sent to the slave computer.

```
Serial::Serial(const char *portName)
{
    this->connected = false; //Try to connect to the given port through CreateFile
    WCHAR wszClassName[1024];
    memset(wszClassName, 0, sizeof(wszClassName));
    MultiByteToWideChar(CP_ACP, 0, portName, strlen(portName) + 1,
        wszClassName, sizeof(wszClassName) / sizeof(wszClassName[0]));
    this->hSerial = CreateFile(wszClassName, GENERIC_READ | GENERIC_WRITE, 0, NULL,
        OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    //Check if the connection was successful
    if (this->hSerial == INVALID_HANDLE_VALUE) {
        //If not success full display an Error... details removed
    }
    else {
        //If connected we try to set the comm parameters... details removed
    }
}

bool Serial::WriteData(const char *buffer, unsigned int nbChar)
{
    DWORD bytesSend; WriteFile(this->hSerial, (void *)buffer, nbChar, &bytesSend, 0);
}

bool Release_Q() //Place down up the cube
{
    bool readResult = SP->WriteData("AAAAAAAAAA", 10); Sleep(60000); return readResult;
}

bool Catch_Q() //Pick up the cube
{
    bool readResult = SP->WriteData("BBBBBBBBBB", 10); Sleep(7500); return readResult;
}
```

Table 5 Simple functions related to serial port communication in Windows

The detailed information, including syntaxs and examples, about how to handle serial port in Windows environment can be found in the links shown below:

[https://msdn.microsoft.com/en-us/library/windows/desktop/aa363858\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa363858(v=vs.85).aspx)

[https://msdn.microsoft.com/en-us/library/windows/desktop/aa365747\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa365747(v=vs.85).aspx)

Slave Computer Side

Besides the sending functions implemented by master computer, the slave computer (the Arduino board) should accounts for receiving the commands, decoding them and controlling the valve signal. In order to meet such requirement, the Arduino board should open the serial port, receive the command and forward it to the optical coupler in the form of voltage. By connecting the Arduino board and the laptop with a USB-UART wire, signals sent from the master computer will be stored in a buffer. Using the function *Serial.read()* provided in Arduino IDE, the program on Arduino board can read the buffer and recognize the useful message in the signals. A simple program shown in **Table 6** can fulfill this task.

```
using namespace std;
void setup()
{
    pinMode(3, OUTPUT); //initialize
    pinMode(2, OUTPUT);
    digitalWrite(3, HIGH);
    digitalWrite(2, HIGH);
    Serial.begin(9600); // set up the serial port
    while (Serial.read() >= 0) {}
}
int NA, NB, i;
void loop()
{
    String tmp;
    char T;
    tmp = "";
    while (Serial.available() > 0) { T = Serial.read(); tmp += T; delay(50); }
    delay(500);
    if (tmp.length() != 0)
    {
        for (i = 0; i < tmp.length(); i++) { NA += (tmp[i] == 'A'); NB += (tmp[i] == 'B'); }
        if (NB > 9)
        {
            digitalWrite(3, LOW); digitalWrite(2, HIGH); // pick up the cube
        }
        else
        if (NA > 9)
        {
            for (i = 0; i < 300; i++) // make sure the cube has been dropped.
            {
                digitalWrite(2, LOW); digitalWrite(3, HIGH);
                delay(100);
                digitalWrite(2, HIGH);
                delay(100);
            }
        }
    }
}
```

Table 6 Functions related to serial port control on Arduino board

Computer Vision System: Setting Up

This subsection shows how we calibrate the wide angle 720p camera and configure the program project appropriately so that it can be compiled with OpenCV.

Calibration of the Wide-Angle Camera

Camera calibration is the process of finding the true parameters of the camera that captures the image. Some of these parameters are focal length, format size, principal point, and lens distortion. These parameters are critical for the wide-angle camera, since wide angle camera introduces a lot of radial distortion to images. Due to radial distortion, straight lines will appear curved. All the expected straight lines are bulged out. An example can be noticed in the **Figure 14**. After calibration processing, the image captured will look normal.

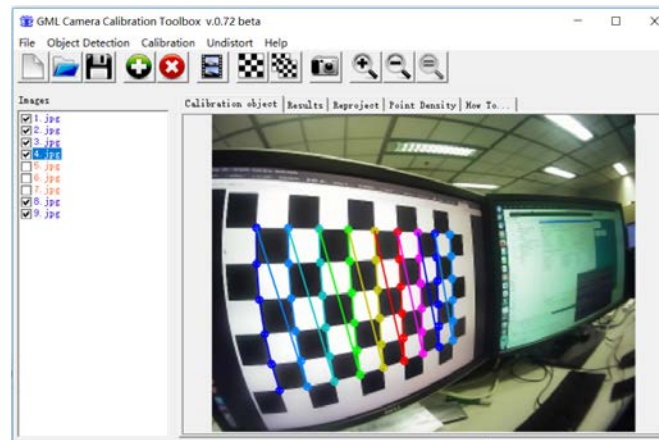


Figure 14 The calibration of camera

Usually, computer vision programmers use chess board to calibrate the camera. With calibration tool, for example, GML which we apply in this lab section, we can get the calibration data and **Figure 15** shows the calibration data of the wide-angle camera that we use in lab section.

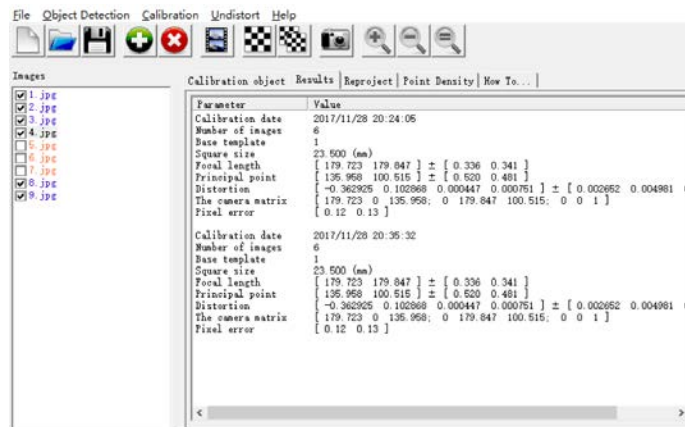


Figure 15 The calibration data of the camera that we use

With the parameters, we can implement the adjustment of the image with OpenCV to get the normal view of the image with a function called *initUndistortRectifyMap* in OpenCV, which remap the image matrix according to the calibration data.

Configuration of Project for OpenCV

The configuration of project for OpenCV is usually our headache but after the lab section, we have understood such procedure clearly.

The OpenCV libraries in Windows are in a Dynamic Linked Libraries (DLL). Another approach is to use static libraries that have lib extensions. In this lab section, both approaches are involved as shown in **Figure 16 & 17**, in other word, we need to not only include the header files and link the libraries, containing the functions or data structures of OpenCV as shown in **Figure 16**, but also inform the executable file where it can find the DLLs, by setting the environment variable *Path* according to **Figure 17**.

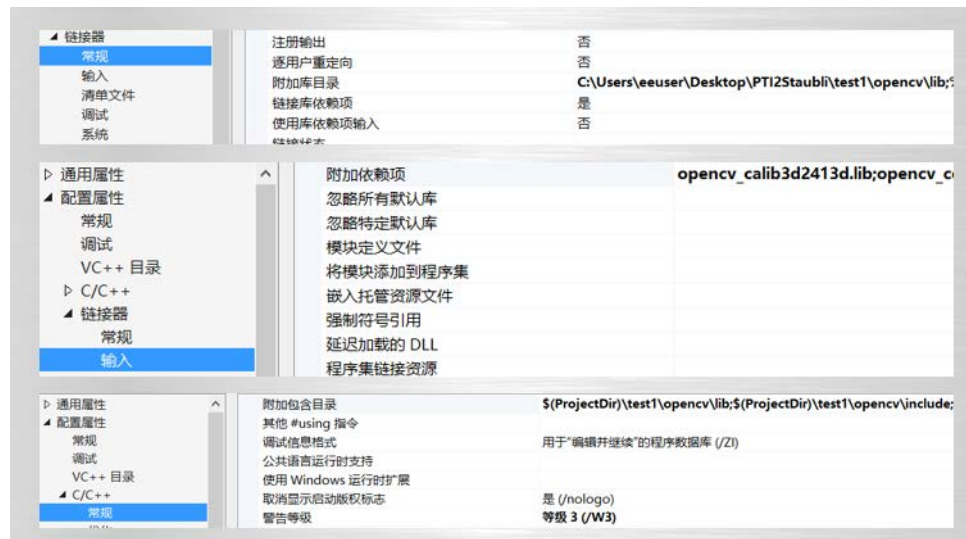


Figure 16 Project Configuration for OpenCV: Header Files and Libraries

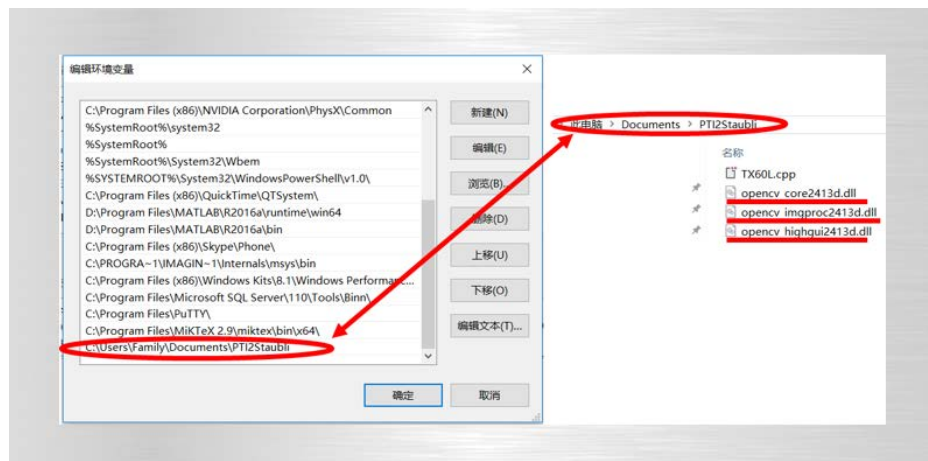


Figure 17 Setting Environment Variable *Path*

After setting up the configuration shown above, the OpenCV project will be ready to be compiled and run.

Computer Vision System: Recognition of Target Cube

This subsection is mainly for illustrating the algorithm we choose and design for the recognition of the target cube. We first analyze the characteristic of the target, choose the basic recognition algorithm appropriately and finally optimize the algorithm to get higher accuracy of recognition.

Characteristics of Target Cube

Before we choose recognition algorithm, it is important to clarify the characteristics of the target cube and these characteristics are listed as below:

- The solid color cube has no texture and patterns,
- The surface of the cube can lead to mirror reflection,
- The size of the cube is small;

The impact of these characteristics is that traditional algorithm based on characteristic points like SURF and SIFT will not be sensitive to the cube and they are not suitable for the detection. Another challenge for cube detection is that, due to the mirror reflection on the surface of the cube as shown in **Figure 18**, the color of the surface could be changed according to the cube position relative to the camera, which means that the algorithm simply based on color detection might turn out to be unstable solution. The small size of the cube might cause the inaccurate positioning when the distance between the camera and the cube is long.

Considering the impacts and challenges discussed above, we first propose a color-based detection algorithm and evaluate its performance.

Basic Recognition Algorithm: A Color-Based Detection Algorithm

Since the most significant feature of the cube is that its color is red, this color-based algorithm is designed to find the largest area in the selected color range in the captured image and position the center of the area as shown in **Figure 18(b)**. In this algorithm, user need to mark the color of the target cube before target positioning. As shown in **Figure 18(a)**, user need to click on the cube in the display window to select the range of the color of the cube. After clicks on the cube, if the user presses the “Y” on the keyboard, the range of color will be recorded for the later target positioning. In contrast, if the “N” on the keyboard is pressed, the range of color selected will be cleared and reset so that user can re-start the procedure of target marking.

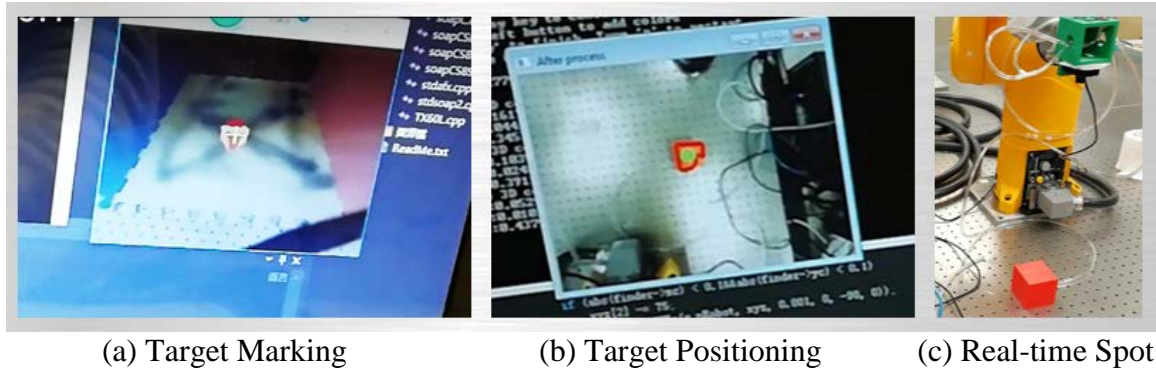


Figure 18 Color-Based Detection Algorithm

In this algorithm, several functions from OpenCV libraries play instrumental roles and they are listed in **Table 7** in the execution order.

Execution Order	OpenCV Function	Comments
1	inRange	Find the areas covered by the selected range of color
2	erode	Noise filter
3	dilate	
4	dilate	
5	erode	
6	threshold	Generate a binary image
7	findContours	Get the contours of the areas
8	moments	Get the size of each contour, find the largest one and return its center.

Table 7 The Execution Flow of the Color-Based Algorithm

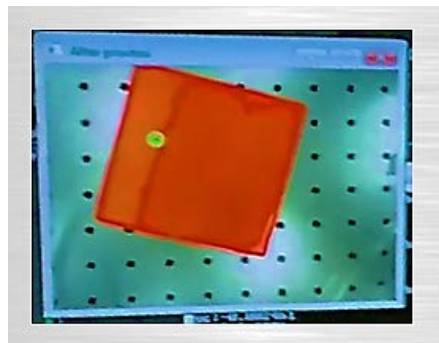


Figure 19 Drawbacks of Color-Based Detection Algorithm

However, due to the color change on the surface of the cube, the detection of the area covered by the selected range of color will lead to unstable result. **Figure 19** shows an failure example where the yellow point is the center of the result area given by the color-

based algorithm and the deviation between the yellow point and the center of the square is significant.

Optimized Recognition Algorithm: Hybrid Detection Algorithm

To overcome the obstacle faced by the color-based algorithm, we combine the edge-based detection algorithm with the color-based algorithm, since as demonstrated in Figure 19, although the color-based algorithm gives the wrong answer, the outline of the cube is quite clear and edge-based detection algorithm can handle this well. A comparison between the two algorithm is demonstrated in **Figure 20**, where the naïve algorithm has lost the target totally while the hybrid algorithm can exactly position the cube and draw the outline of it.

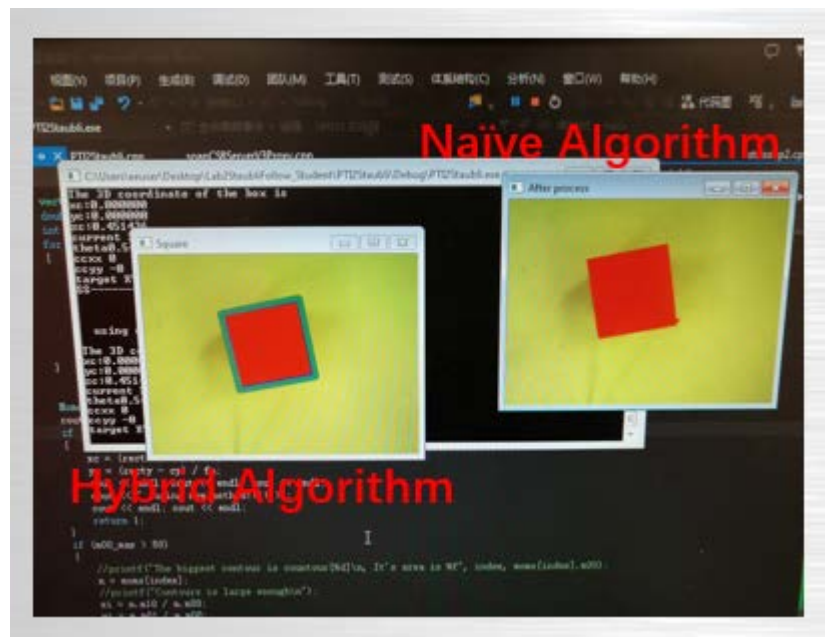


Figure 20 Comparison between Hybrid Algorithm and Naïve Color-Based Algorithm

With the help of edge detection algorithm, we can easily get the polygons, including squares, in the captured image. The edge-based detection algorithm is not sensitive to the change of color. However, if the camera is far from the target cube, the cube could be too small to be detected as a square and the color-based algorithm can make up such disadvantage of the edge-based detection algorithm.

To implement the edge-based detection algorithm, two functions from the libraries of OpenCV are important. One is *cvFindContours* and the other one is *cvApproxPoly*. *cvFindContours* can extract the contours, which could be polygons potentially, from the image and store them in a sequence. Then, *cvApproxPoly* will be applied to the elements in the contours sequence and it will approximate the curves of these contours. Finally, simple checks on the angles, the areas and the portion of edges will point out the target square in the image.

The details of the hybrid algorithm are described in **Figure 21**. According to **Figure 21**, For each iteration of the hybrid algorithm, both edge-based detection algorithm and color-based algorithm will be executed and output their results, containing the area and center of the target detected. The range of the area of the roof surface in the image can be estimated according to the height of the end-effector. Based on the estimated range of area, we can make the decision between the results from the two algorithms. If the result area given by color-based algorithm is in the estimated range, the result of color-based algorithm will be accepted. Otherwise, the result of edge-based detection algorithm will be evaluated and if its area is in the estimated range, its result center of the cube will be adopted. If both of them have the out-of-range result, the hybrid algorithm will report the lost of target and the robot arm might sweep to try to find the target.

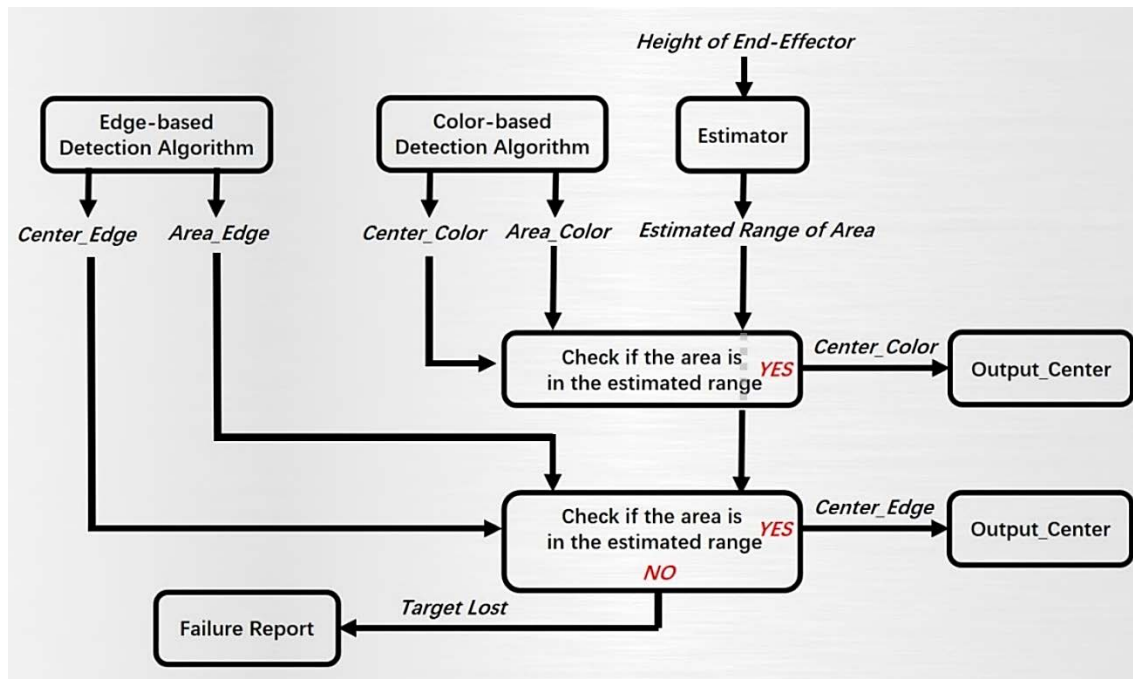


Figure 21 Hybrid Detection Algorithm

Computer Vision System: Recognition of Box

This subsection is mainly for illustrating the algorithm we choose and design for the recognition of the box, where the target cube is located.

Compared to the target cube, the characteristics of the box are relatively significant:

- The inner surface is the combination of several solid color blocks,
- The area of each color block is relatively large,
- The relative positions of the color blocks are certain;

An example is shown in **Figure 22** to illustrate the characteristics of the box. As shown in **Figure 22**, three color blocks are visible to the camera. Block 1 is blue, block 2 is yellow and block 3 is red. Their IDs indicate their positions from left to right.

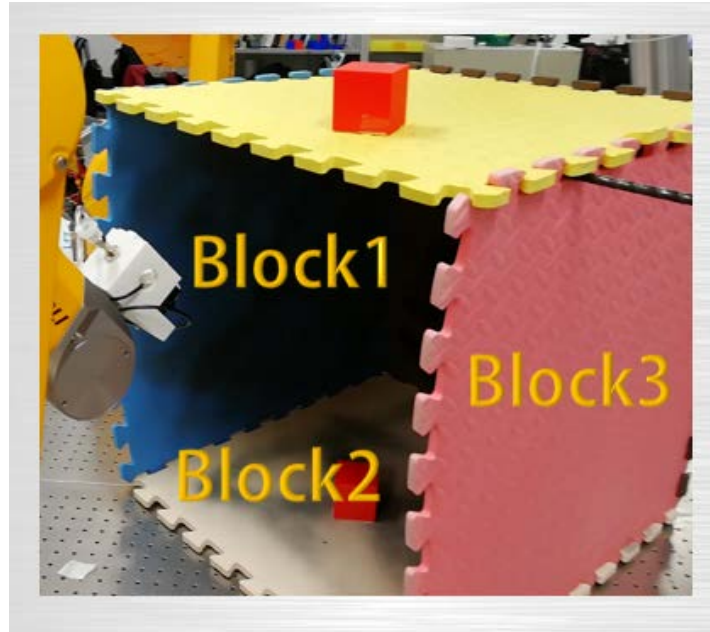


Figure 22 An Example of the Box

Therefore, the colors and the relative position of the three blocks can be marked in advance and the procedure of marking is demonstrated in **Figure 23**. When the task is started, the robot arm will sweep around itself to find the appropriate combination of the three color blocks and position the box.



(a) Mark Block 1

(b) Mark Block 2

(c) Mark Block 3

Figure 23 Marking of the Box



Figure 24 Detection of the Box

When the camera detects three large color blocks and they are in right position like the example shown in **Figure 24**, it means the box is in front of the camera and the robot arm will stop sweeping and starting the recognition of the target cube.

Section IV. Strategies to Finish the Task

As mentioned in Section I, we divide the task into six steps. For each step, different strategies are involved. Based on the relative low-level designs in subsystem illustrated in section III, master computer system can combine them to be the solution to each step. In this section, these strategies will be illustrated in the order of execution. To make the explanation clear, here we reuse **Figure 3** in Section I and regard it as **Figure 25**.

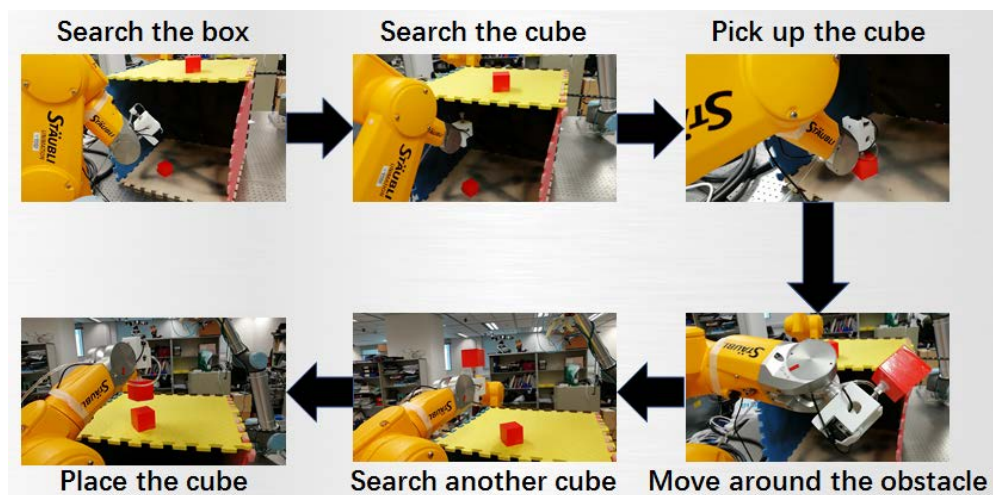


Figure 25 Execution flow of the lab task

Step I: Search for the Box

As the first shown in **Figure 25**, since the location of box is unknown in advance, the robot arm will sweep around itself. As shown in **Figure 26**, the base of the robot arm will be rotated a small angle and the camera will capture an image for the detection algorithm to check whether the box is in front of the camera. The rotation will be continued until the box is detected. What should be noticed is that during the sweep, the camera does not aim downward. Instead, it pitches up a little to extent the search scope of box as shown in **Figure 27**.

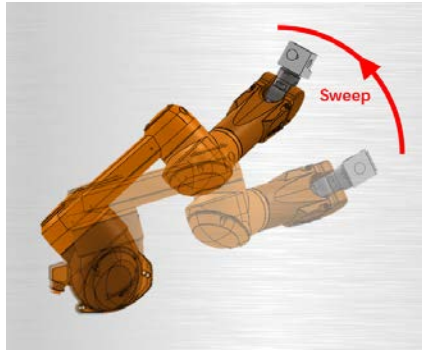


Figure 26 Sweep to Search

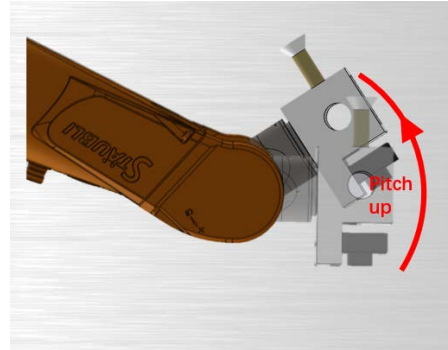


Figure 27 Pitching up of the Camera

Step II: Search for the Cube

When the camera detects the box, the end-effector will turn to be leveled so that the camera will aim downward and be ready to search for the cube. If the cube is not in the scope of the camera, the end-effector will sweep in the box to find the cube. When the cube is in the scope of camera, the end-effector will try to move to the position right above the cube to let the cube locate in the center of the image. To reach this goal, we need to transform the deviation vector between the center of image and the position of cube in image into the body frame so that the robot arm will know what motion it should accomplish.

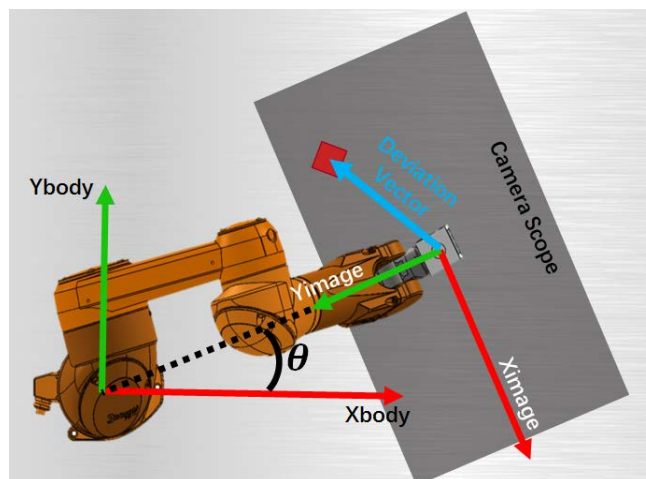


Figure 28 Transformation of Deviation Vector from Image Frame to Body Frame

According to the transformation matrix between camera frame and body frame of the robot arm, we get expression of deviation vector in body frame:

$$\begin{cases} \theta = \text{atan2}(y_{\text{end-effector}}, x_{\text{end-effector}}) \\ dx_{\text{body}} = K(\text{Height}) \times (\sin\theta \times x_{\text{image}} - \cos\theta \times y_{\text{image}}) \\ dy_{\text{body}} = K(\text{Height}) \times (-\cos\theta \times x_{\text{image}} - \sin\theta \times y_{\text{image}}) \end{cases}$$

$x_{\text{end-effector}}$ and $y_{\text{end-effector}}$ is the end-effector ordinate in body frame. Based on this ordinate, we obtain the relative angle between image frame and body frame, θ , as shown in **Figure 28**. $(x_{\text{image}}, y_{\text{image}})$ is the cube ordinate in image frame. $(dx_{\text{body}}, dy_{\text{body}})$ is the deviation vector in body frame. $K(\text{Height})$ is a relative coefficient, a function of the height of the end-effector, which is applied because the size of camera scope will be changed according to the height of end-effector.

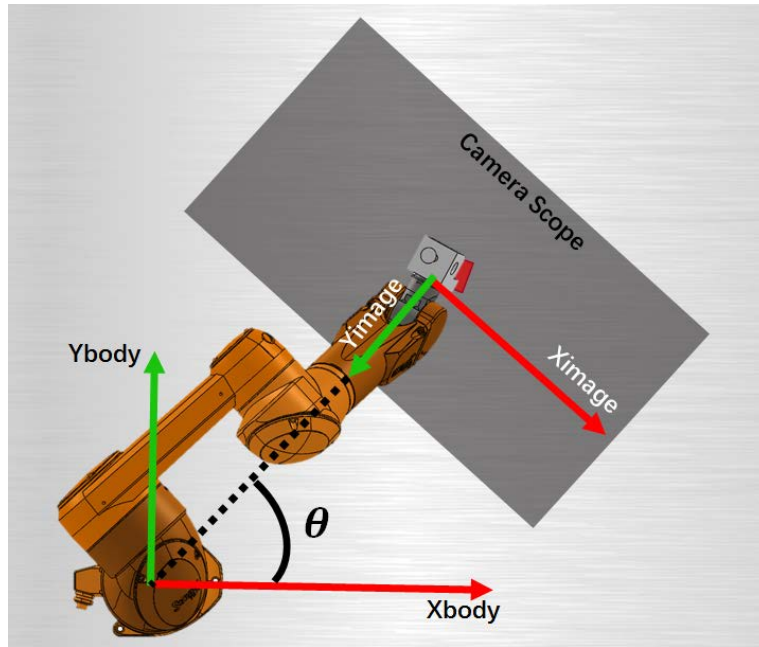


Figure 29 The end-effector reaches the target position.

According to the deviation vector, the target position of end-effector can be obtained:

$$\begin{cases} x_{\text{target}} = x_{\text{end-effector}} + dx_{\text{body}} \\ y_{\text{target}} = y_{\text{end-effector}} + dy_{\text{body}} \end{cases}$$

In this way, the end-effector will be led to the position right above the cube as shown in **Figure 29**. After that, the robot arm will lower the height of the end-effector so that the end-effector can get closer to the cube. A loop of image capturing, target detection and end-effector motion will be continued until the end-effector get close enough to the cube so that it can pick up the cube accurately.

Step III: Pick Up the Cube

As shown in **Figure 30**, when the camera gets close enough to the cube, the end-effector will rotate 180 degrees and the suction pad will aim downward. Then, the end-effector will decline and the suction pad will touch on the tube. After that, the master computer system sends the PICK command to the controller of pneumatic system, the Arduino board, and the cube will be attached to the suction pad. Finally, the end-effector will rotate 180 degrees, the cube will be on the end-effector roof and the camera will restore the status of aiming downward so that it is ready for searching for another target cube.

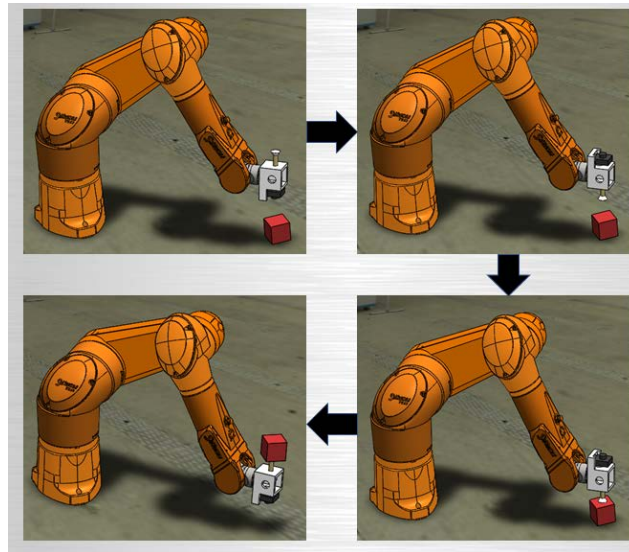


Figure 30 The Procedure of Picking Up the Cube

Step IV: Move around the Obstacle

After picking up the cube, the end-effector needs to move around the obstacle, the box, as shown in **Figure 31**.

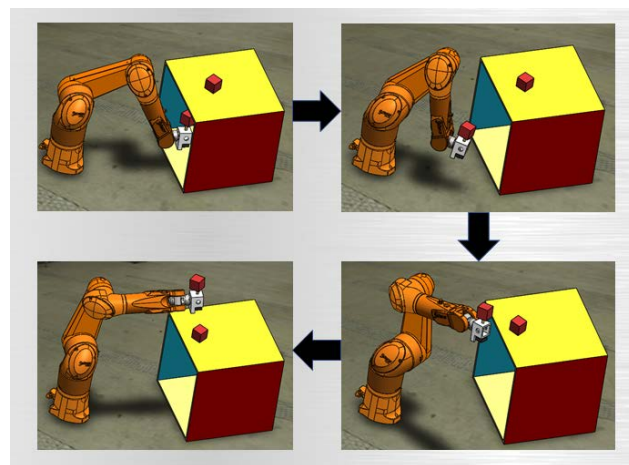


Figure 31 The Procedure of Moving around the Box

Firstly, the robot arm needs to be withdrawn toward its base so that the end-effector will move out of the box. Secondly, the height of the end-effector will uplift over the roof of the box. Thirdly, the robot arm turns around to move the end-effector to the position over the roof of the box and the camera will be ready for searching for another cube.

Step V: Search for another Cube

This step is similar to the step II. The major difference is that since the camera will be close to the roof and its scope will be quite limited. As result, the end-effector need to sweep a relatively large area to find the other cube and position it.

Step VI: Place the Cube

As the final step of the task, after the positioning of the other cube, the end-effector will rotate 180 degrees and the suction pad will aim downward. the master computer system sends the PLACE command to the controller of pneumatic system, the Arduino board, and the cube will be released from the suction pad. Such procedure of placing the cube is depicted in **Figure 32**.

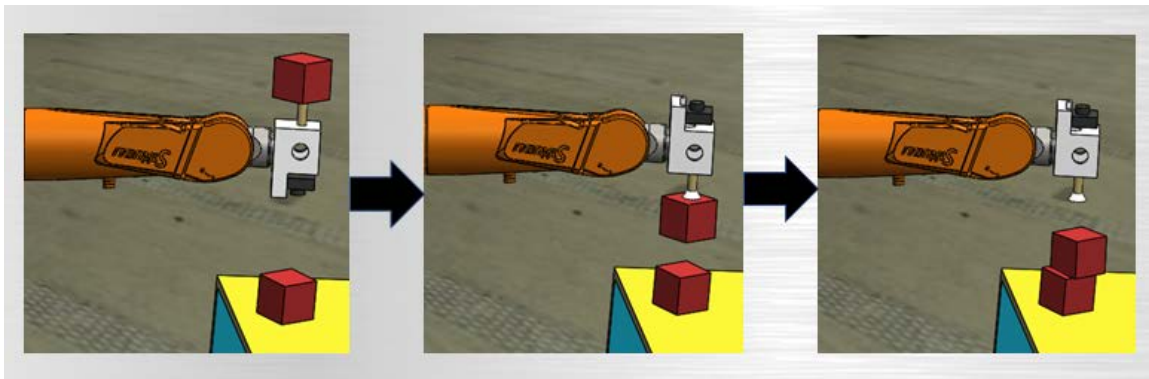


Figure 32 The Procedure of Placing Down the Cube

Acknowledge

We sincerely appreciate Professor Zexiang LI for his detailed and patient illustration about robotics in this semester and it is truly the brand-new start point of our exploration of robot. We are also grateful for the time and the effort paid by T.A Di ZHANG and T.A. Lixiang LIAN. Thanks to their patience, we can understand the interesting parts in practicing in real applications. Finally, we are so lucky to have the chance to share the wonderful time with nice people in Robotics Institute of HKUST!