102-1 Robotics Final Project Report

Clerk Robot

Group 1: 林叔君(Shu-Chun Lin), 趙冠琳(Guan-Lin Chao), 陳威宇(Wei-Yu Chen)
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
{ r02921013, b99901164, b00901079 }@ntu.edu.tw

***Abstract*—This project is about the clerk robot which can keep customers' valuable belongings temporarily. The clerk robot is constructed by one UAL5D arm, one pioneer and one Kinect.**

Keywords*—Kinect; UAL5D; Pioneer; skeleton; calibration; digit recognition; inverse kinematics

I. INTRODUCTION

We want to build a clerk robot so that the customers can deposit or withdraw their belongings. It is constructed by one UAL5D arm, one Pioneer and one Kinect to achieve the hand, legs and eyes function respectively. For the Kinect, eyes, we use it to locate the customer, position of the grip of the box so that the Pioneer and the UAL5D arm know that where they should go. What's more, to do the digit recognition, we need the images captured by the Kinect. For the Pioneer, legs, we use it to move to the assigned position, customer and the shelves. As for the UAL5D arm, we use it to grip the box and hand it to the customer when the customers want to deposit or withdraw.

II. SYSTEM OVERVIEW

Fig. 1 is the block diagram of our scenario. Individual modules are represented by boxes of different filling color. Orange boxes correspond to Kinect's image recognition; cyan boxes correspond to Pioneer's motion control; red boxes correspond to Speech recognizer's keyword spotting; purple boxes correspond to Text-to-speech modules; green boxes correspond to UAL5D robot arm manipulation; pink boxes correspond to number recognition module; blue boxes correspond to other computations.

III. LOCATING THE CUSTOMER

The NiTE2 library of OpenNI 2 provides a nite::UserTracker class of human tracking. The objects of this class can manage human users, track skeletons, detect postures. Besides nite::UserTracker, objects of nite::HandTracker class can detect hand positions and recognize hand gestures.

There are several member functions of significant use:

- getUsers(): Get the list of current users.
- startSkeletonTracking(): Start tracking the skeleton of a specific user.
- stopSkeletonTracking(): Stop tracking the skeleton of a specific user.
- nite:: Skeleton getSkeleton: return the skeleton of a specific user
- nite::SkeletonJoint getJoint(): return a specific joint data of the skeleton

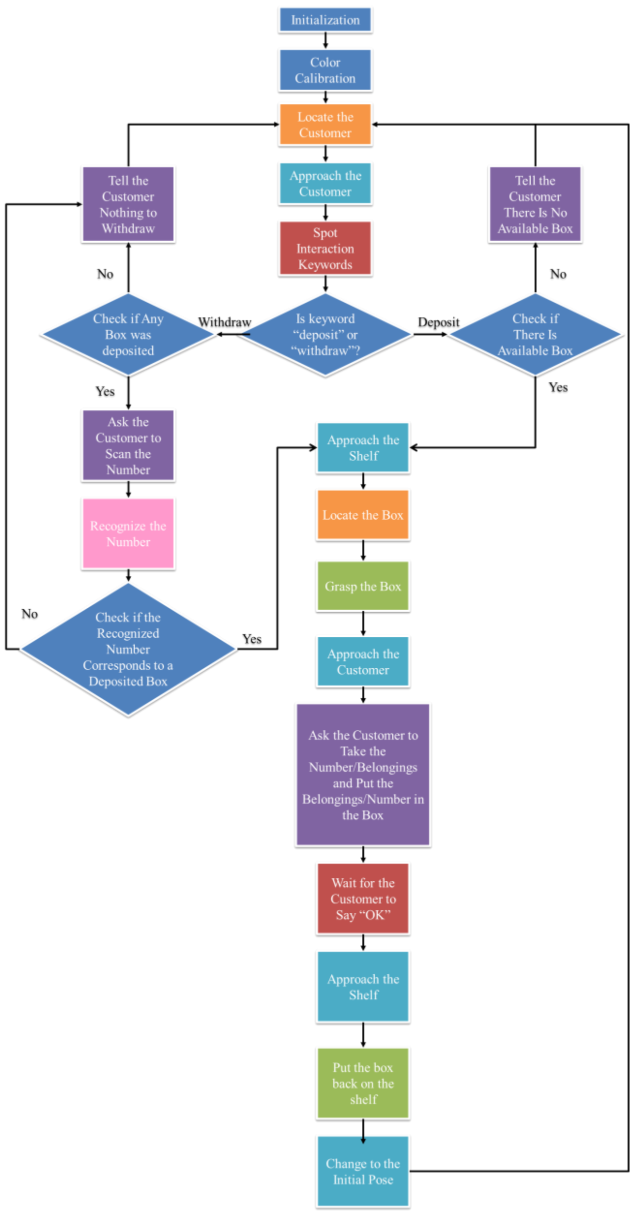
Initialization
Color Calibration
Locate the Customer
Approach the Customer
Spot Interaction Keywords
Tell the Customer Nothing to Withdraw
Tell the Customer There Is No Available Box
No
No
Check if Any Box was deposited
Withdraw
Is keyword "deposit" or "withdraw"?
Deposit
Check if There Is Available Box
Yes
Yes
Ask the Customer to Scan the Number
Approach the Shelf
Recognize the Number
Locate the Box
Grasp the Box
No
Check if the Recognized Number Corresponds to a Deposited Box
Yes
Approach the Customer
Ask the Customer to Take the Number/Belongings and Put the Belongings/Number in the Box
Wait for the Customer to Say "OK"
Approach the Shelf
Put the box back on the shelf
Change to the Initial Pose

Fig. 1. System Overview

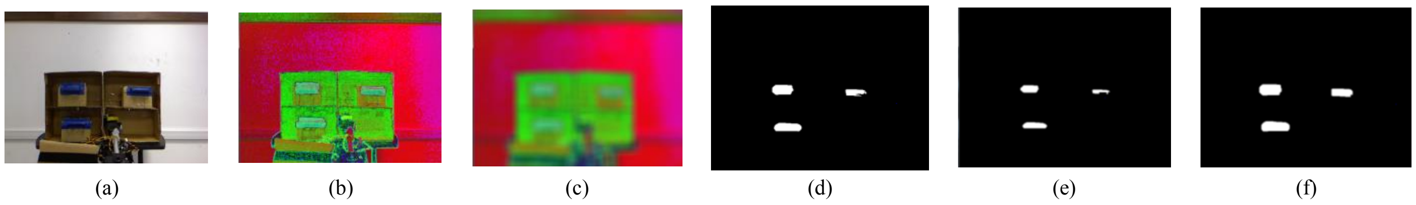

Fig. 3. The procedures (a) raw image in RGB space, (b) raw image in HSV space, (c) blur, (d) range, (e)erosion, (f) dilation.

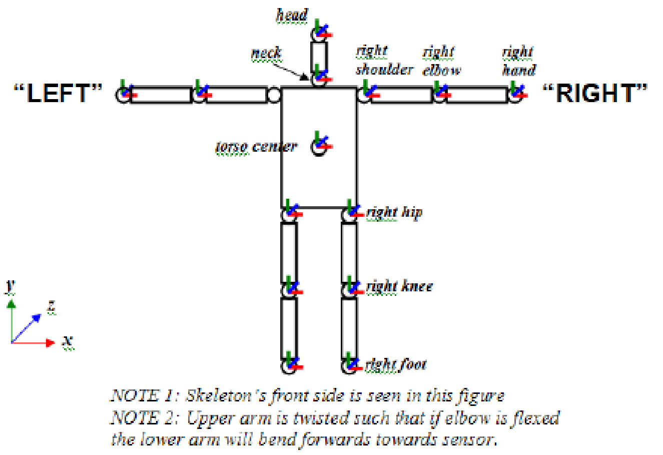
head
neck
right shoulder
right elbow
right hand
"LEFT"
"RIGHT"
torso center
right hip
right knee
right foot
y
z
x
NOTE 1: Skeleton's front side is seen in this figure
NOTE 2: Upper arm is twisted such that if elbow is flexed the lower arm will bend forwards towards sensor.

Fig. 2. Joints are (1)JOINT_HEAD, (2)JOINT_NECK, (3)JOINT_LEFT_SHOULDER, (4)JOINT_RIGHT_SHOULDER, (5)JOINT_LEFT_ELBOW, (6)JOINT_RIGHT_ELBOW, (7)JOINT_LEFT_HAND, (8) JOINT_RIGHT_HAND, (9)JOINT_TORSO, (10)JOINT_LEFT_HIP, (11)JOINT_RIGHT_HIP, (12)JOINT_LEFT_KNEE, (13)JOINT_RIGHT_KNEE, (14)JOINT_LEFT_FOOT, (15)JOINT_RIGHT_FOOT

There are 15 predefined joint types, as shown in Fig. 2

We tracked the head, right hand, and left hand positions of the user. When the user raises his hand to greet the robot (y coordinate of right hand or left hand higher than head), the system returns the current position (here, we used head position to represent the user position) of the user. It is interesting to note that the OpenNI utilizes a left-hand coordinate system. We need to transform it to our world coordinate system with caution.

IV. Locating the Handle Positions of the Box

By calibrating the color range in the initialization of our scenario, the color range of box handles were calibrated to a feasible range. The raw RGB images are then transformed to HSV space, blurred, ranged, eroded, dilated to make the handles connected components in the image. Then we calculate the centroid of each connected components and use them as the position of the handles. The procedures are shown in Fig. 3 (a) ~ (f)

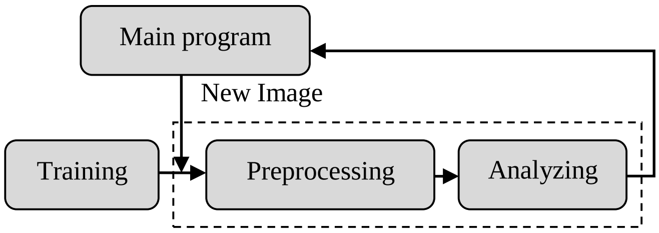

Fig. 4. Flow Chart of the Digit Recognition

V. Digit Recognition

The flow chart of digit recognition is shown in Fig. 4. The first thing we have to do is to train the digits. We use the K-nearest to train 0~9 at the first. After training the digits, we have to wait for the main program to capture a new frame and call the digit recognition function. Once the main program get into the digit recognition function with a new image from Kinect, we have to do some preprocess to find the area of the digit correctly, that is, with right bounding boxes. In order to find the right bounding boxes of the digit, we use a color filter to filter out the red region. The filter is designed by using the HSV color space. However, since the HSV value of the same color will be changed with lightness, we have to use the color calibration so that we can find the right upper and lower bound to filter out the red region of the digit. To remove the small areas, we also use erosion and dilation to remove the noise. After preprocessing, we analyze the image with bounding boxes and trained result using the K-nearest method to get the digit. Finally, it will end this function, go back to the main program and wait for the main program to call it.

VI. Moving the Pioneer

The control of pioneer is actually based on functions offered by ARIA, and we can directly set its position rather than control its speed. As illustrated in Fig. 5, at the beginning of our scenario, our robot stands in front of the shelf first, and sets this point as the origin of the world frame.

After color calibration, it turns around and search for user as mentioned in scenario description. Once find a user, it will approach him or her. Then it turns back to origin in the grasping part. Limited error is acceptable, which we'll mention it in finding part, but to enhance the precision of robot's position, we let pioneer move along x and y direction only.

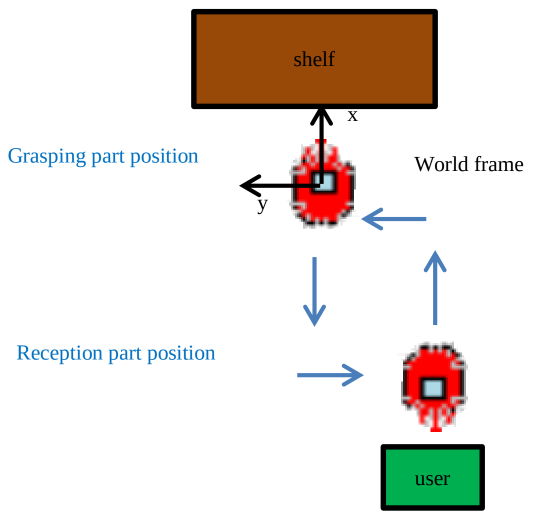

Fig. 5. Pioneer movement

VII. MANIPULATING THE UAL5D ARM

In our scenario, there are two functions we need to achieve through arm control: grasping the assigned box and putting it back on the shelf. We use the UAL5D arm to handle this part, which has the following D-H Representation as illustrated in table I and Fig. 6.

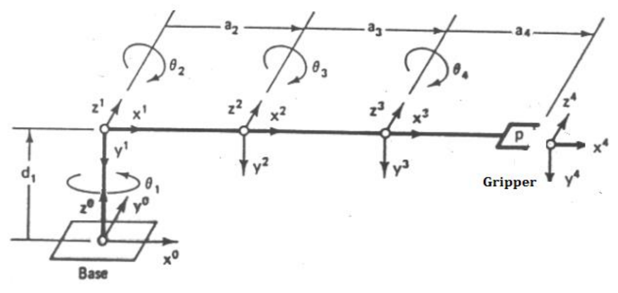

Fig. 6. Link configuration

TABLE I. D-H REPRESENTATION

Axis	θ	d	a	α	Home
1	θ1	7	0	-π/2	0
2	θ2	0	14.6	0	0
3	θ3	0	18.5	0	0
4	θ4	0	7.1	0	0

After some adjustment, these joint angles can be manipulated by program through RS232 port. To grasp the box precisely, we need to solve the arm's inverse kinematics and then is able to control its position. Since we require the gripper approaching the handle of the box horizontally, θ2 +θ3 +θ4 should equal to zero. Set

$$L = \sqrt{x^2 + y^2} - a_3, H = z - d_1,$$
$$R = \sqrt{H^2 + L^2}, \varphi = \operatorname{atan2}(H, L)$$

Then we have:

$$\theta_1 = \operatorname{atan2}(y, x)$$
$$\theta_2 = -\cos^{-1}\left(\frac{R^2 + {a_2}^2 - {a_3}^2}{2Ra_2}\right) - \varphi$$
$$\theta_4 = -\cos^{-1}\left(\frac{R^2 + {a_3}^2 - {a_2}^2}{2Ra_3}\right) + \varphi$$
$$\theta_3 = -\theta_2 - \theta_4$$

With the position information offered by camera, we can grasp the assigned box successfully. The gripper keeps gripping the box till putting it back, where the arm simply move in reverse order of grasping part, rather than calculating the position of shelf once again.

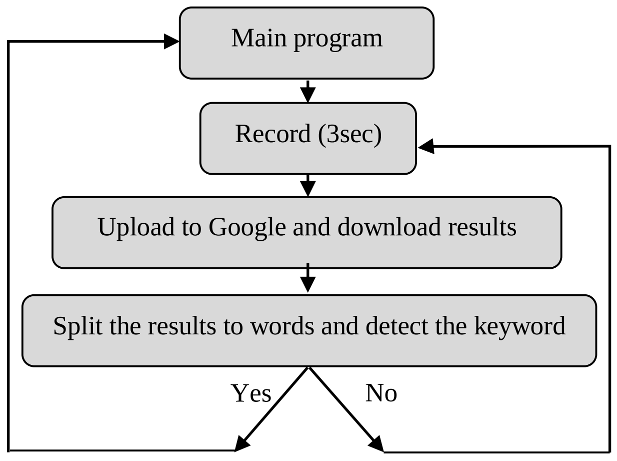
Main program
Record (3sec)
Upload to Google and download results
Split the results to words and detect the keyword
Yes
No

Fig. 7. Flow Chart of Speech Recognition

VIII. SPEECH RECOGNITION

The flow chart of speech recognition is shown in the Fig. 7. We write the record function and then using the existing material to upload the sound to Google speech recognition and download the results. After that, we also write the function to split the results to single words and detect that whether there is the keyword. It will end this function until it detect the keyword, or it will keep record the sound and detect the keyword.

IX. FINDINGS

A. *Coordinate Transformation from Kinect Depth Image to World Coordinate*

We utilized a pinhole camera model to transform the Kinect depth images to world coordinate in Fig. 8:

x-axis focal length $f_x \approx 525.0$ pixel
y-axis focal length $f_y \approx 525.0$ pixel
x coordinate of principal point $x_c \approx 319.5$ pixel
y coordinate of principal point $y_c \approx 239.5$ pixel
world coordinate $z_w = \text{depth}$
world coordinate $x_w \approx \frac{(x - x_c)}{f_x} \cdot z_w$
world coordinate $y_w \approx \frac{(y - y_c)}{f_y} \cdot z_w$

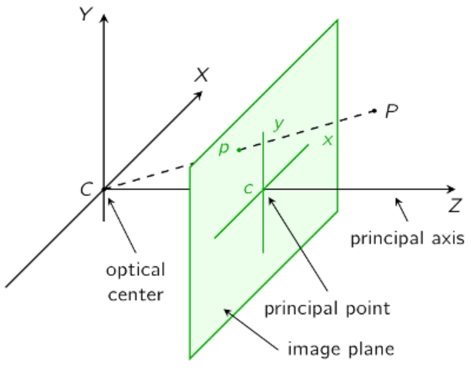

Fig. 8. Pinhole camera model

B. Color Space Issues

We initially used RGB color space to calibrate and range images to locate the blue box handles. However, it suffered from severe background noises and lighting changes of the environment. So we converted the color to HSV space, which was more robust to lighting changes.

HSV is a cylindrical-coordinate representation of color space, as shown in Fig. 9. HSV stands for hue, saturation, and value, and is also often called HSB (B for brightness). In OpenCV, the color conversion between RGB and HSV is shown by the following formula:

In case of 8-bit and 16-bit images, R, G, and B are converted to the floating-point format and scaled to fit the 0 to 1 range.

$$V \leftarrow \max(R, G, B)$$

$$S \leftarrow \begin{cases} \dfrac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} \dfrac{60(G - B)}{V - \min(R, G, B)} & \text{if } V = R \\ 120 + \dfrac{60(B - R)}{V - \min(R, G, B)} & \text{if } V = G \\ 240 + \dfrac{60(R - G)}{V - \min(R, G, B)} & \text{if } V = B \end{cases}$$

If $H < 0$, then $H \leftarrow H + 360$. On output$0 \leq V \leq 1, 0 \leq S \leq 1, 0 \leq H \leq 360$.

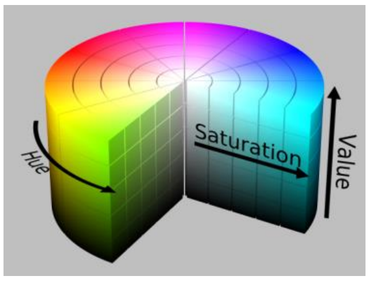

Fig. 9. HSV color space

C. UAL5D arm

There are some none-ideal effects we need to handle, which we'll mention it in following part.

1) Gravity Compensation:

Due to the influence of gravity, the gripper's z position will be lower than theoretical value when arm stretching out. We find that adjusting θ_2 and θ_3 about 5 degrees can approximately remove the bias, while it may lead to errors once arm draws back. In fact, we need precise position during grasping procedure only, so such compensation is feasible.

2) Precision issue:

Even with gravity compensation, there are still some errors due to the camera, pioneer movement or difference between theoretical and exact joint angle. After observing the shelf as illustrated in Fig. 10., we find that limited error in x and y direction is endurable since the handle has its width and depth. However, to grasp the box precisely, the greatest tolerable error in z direction is only 1 cm due to the limit of gripper. To ensure its precision, we trickily fix the arm's z position in grasping part, which is feasible since the height of arm's base and the shelf is fixed.

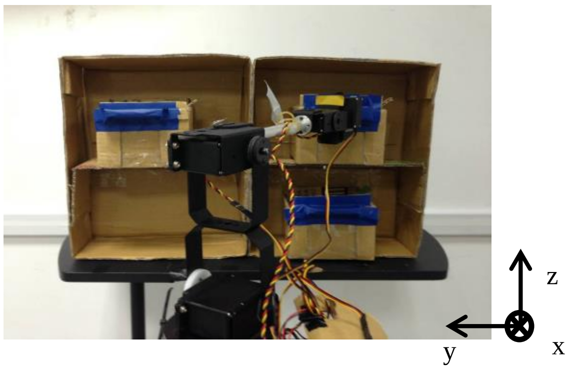

Fig. 10. Scenario of grasping the box

3) Resistance force limitation:

While grasping a box, the weight of box will add force to the robot arm. If we raise it too fast, resistance force will excess the limitation, therefore the arm will immediately hang down and even lose control. To avoid such situation, we define the arm's moving track first. By assigning several goals along the track continuously, the arm can follow it in relevant speed. With such method, even the arm losing control suddenly, it will still move to the next goal we assigned.

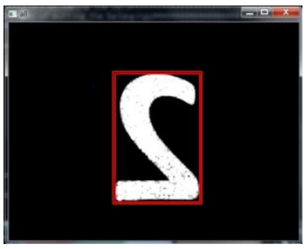

Fig. 11. Digit image after preprocessing

D. Digit Recognition

The images after preprocess are shown in Fig. 11. As you can see, the image after color filter and erosion and dilation will have less hole. This can highly increase the correct rate.

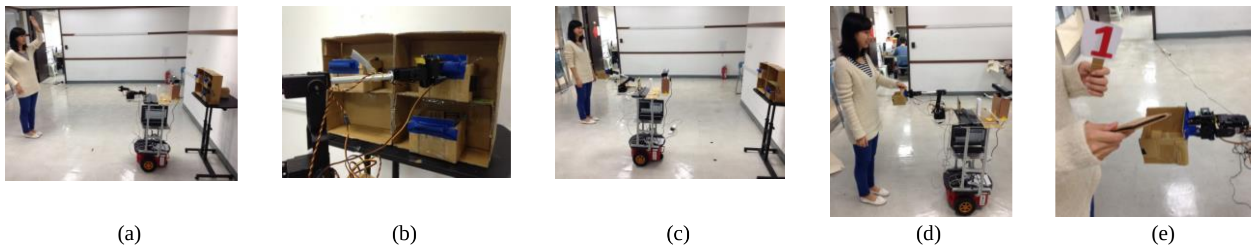

Fig. 12. Scenario, (a) robot located the customer, (b) robot grasped an empty box, (c) robot approached the customer, (d) robot asked the customer to take the number from the box, (e) the customer put the belonging in the box

What's more, we can find the right bounding boxes using the findContour() function so that we can analyze it correctly.

X. CONCLUSION

In this project, we realized six functions, including digit recognition, speech recognition, customer localization, box handles localization, Pioneer control and UAL5D arm control. With the organization of these functions, we can construct a complete reception and inventory management system. Once we enhance its scale and stability, it may serve as a real storage system like Amazon. Fig. 12. is the photos of the user scenario.

XI. WORK DISTRIBUTION

Work	Contributor
Scenario Planning	Guan-Lin Chao
Locating the Customer	Guan-Lin Chao
Locating the Handle Positions of the Boxes	Guan-Lin Chao
Digit Recognition	Shu-Chun Lin
Speech Recognition	Shu-Chun Lin
Moving the Pioneer	Wei-Yu Chen
Manipulating UAL5D Arm	Wei-Yu Chen
Report	Shu-Chun Lin, Guan-Lin Chao, Wei-Yu Chen

XII. REFERENCES

[1] http://viml.nchc.org.tw/blog/paper_info.php?CLASS_ID=1&SUB_ID=1&PAPER_ID=434

[2] http://docs.opencv.org/2.4.7/doc/tutorials/tutorials.html http://pille.iwr.uni-heidelberg.de/~kinect01/doc/reconstruction.htmlI.

[3] https://groups.google.com/forum/#!topic/openkinect/ihfBIY56Is8

[4] http://en.wikipedia.org/wiki/HSL_and_HSV