Ambient Intelligence for Safe Robot Sorting

Zhang Yi[1], Xie SongHua[1],Liu Zijian[1],He ChongShan[1],Zhou Yuheng[1],Wang Huacen[1],Jiang Yuwei[1],Qiu Tian[1],Li Boxuan[1]

***Abstract*— As collaborative robots share workspaces with people, security concerns are growing.In this paper, we study the robot environment perception in garbage sorting process.The camera is used to recognize the human body around the surrounding manipulator, and the operating speed of the manipulator is adjusted in real time according to the result of recognition.For human body detection, SSD and YOLO algorithms are tested, and the two algorithms are compared.When the human body is identified, the distance information of the human body region is calculated, and the speed of the manipulator is adjusted with this reference.In addition, we also collect and label different kinds of garbage data to help the robot achieve better sorting effect.**

***Index Terms*— Ambient Intelligence, Garbage Sorting, Cooperative robot**

Contents

*This work was supported by Southern University of Science and Technology and AncoraSpring Inc. for the ME336 Collaborative Robot Learning (`me336.ancorasir.com`) under the supervision of Prof. Song Chaoyang `songcy@ieee.org`

#Equal contribution as co-first authors.

[1] Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China.

I. Introduction

In traditional industrial robot systems, the working spaces between workers and robots are separated from each other. In recent years, the robotics industry has gradually shifted to collaborative robot systems. This system has a collaborative working space where robots and humans can work in operation at the same time. However, in a collaborative robot system, due to the collaborative work of humans and robots, safety accidents may occur due to accidental collisions. Many papers have proposed safety systems in collaborative robot systems and methods for estimating worker positions for configuring safety systems. According to the international standard ISO/TS 15066 for collaborative robot systems, it is necessary to know the position and speed of workers to ensure their safety when working together. As shown in Figure 1, the entire system includes a waste sorting production line consisting of a conveyor belt and a robotic arm, and a camera for monitoring the working environment of the robotic arm. In this project, we use the camera to detect the working space of the robot. When the human body is detected, the operating speed of the robotic arm will be adjusted in real time according to the distance from the human body to the robot. In addition, we applied this system to a real garbage sorting scene to test its safety perception ability in the sorting process.

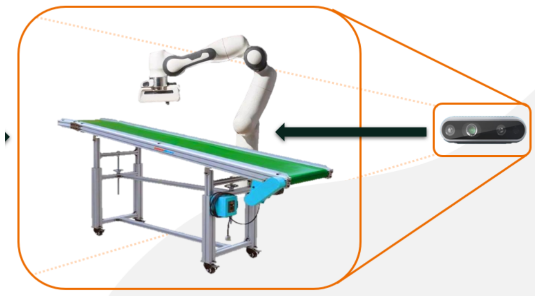


Fig. 1: Schematic diagram of system architecture

A. Related Work

There are some papers on collaborative robot environment perception. In some papers, a monocular camera is used to recognize the human body, and then the distance of the human body is judged according to the parameter matrix of the camera and the size of the human body. The accuracy of this brief method of measuring distance is low, and the algorithm is more complicated. There are also researchers who use lidar to scan the surrounding environment. This method can accurately know the surrounding three-dimensional environment information. However, the cost of lidar is high, and there is no way to distinguish the types of surrounding objects. In this project, we use a depth camera to calculate the distance based on the surrounding point cloud information after recognizing the human body. This method can not only identify people but also get the distance accurately with minimal cost.

B. *Contributions*

System Engineer: Zhang Yi
Algorithm Engineer: Zhou Yuheng, He Chongshan
Data Engineer: Wang HuaCen, Jiang YuWei
Design Engineer: Qiu Tian, Li Boxuan
Software Engineer: Xi SongHua, Liu Zijain

II. Method

A. *Algorithm Solution*

In this chapter, a human detection method by using yolov5 is developed. Safety helmet wearing detect dataset (SHWD) and yolov5s model are applied for the training.

1) Dataset introduction: SHWD provides the dataset used for both safety helmet wearing and human head detection. It includes 7581 images with 9044 human safety helmet wearing objects (positive) and 111514 normal head objects (not wearing or negative). The positive objects got from Google or Baidu, and are manually labeled with LabelImg. Some of the negative objects got from SCUT-HEAD.

2) Training procedure: (i) Prepare the environment of yolov5. (ii) Create the dataset configuration file, which the class is set as “person”, “head”, and “helmet”. (iii) Create a label file for each image. The file includes the information of bounding boxes. (iv) Make sure all the files are placed correct. (v) Choose yolov5s model and modify the class number to 3. (vi) Start the training.

3) Interface design: Due to the fact, that the original code in yolov5 has no interface for the function call, it is required to design a function to call detect.py. Function input: source images/ videos and some format with yolov5 input. Function output: ifhelmet: True for helmet exists and false for helmet absents; rectangles: locations of bounding boxes of helmets or humans, with the format (x1, y1, x2, y2); centers: center locations of helmets or humans, with the format (x, y). In order to test the interface, use an image to get its result as the following figure and the running time to detect one image is average of 0.03 second.

4) SSD Introduction: The SSD model regards the output bounding boxes as a group of boxes of different scales and sizes, and sets a series of default boxes of different sizes on the feature maps of different scales.The implementation steps of the specific method are as follows:

(i)Input a picture and input it into the pre-trained classification network to obtain feature mappings of different sizes;

Fig. 2: Source image

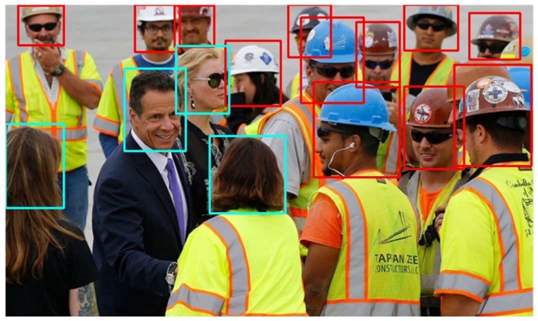


Fig. 3: Result image

(ii)Six layers of feature maps in different size mappings were extracted, and then six bounding boxes of different scales were constructed at each point above these feature map layers, and then multiple bounding boxes were generated through detection and classification.

(iii)The bounding boxes obtained from different feature maps were combined and some overlapping or incorrect bounding boxes were suppressed through non-maximum suppression method to generate the final bounding box set, so as to obtain the detection results.

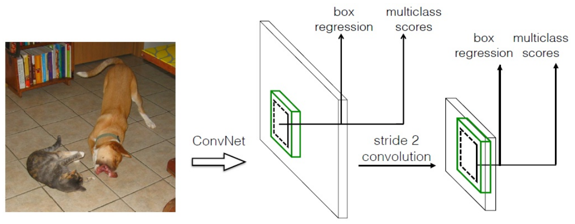


Fig. 4: The learning result

Compared with YOLO, SSD uses CNN to conduct detection directly, instead of testing after the full connection layer like YOLO.In addition, SSD extracts feature maps of different scales for detection. Large scale feature maps (the first feature map) can be used to detect small objects, while small scale feature maps (the second feature map) can be used to detect large objects.At the same time, SSD also uses a priori frame of different scales and aspect ratios.The disadvantage of YOLO algorithm is that it is difficult to detect small targets and inaccurate positioning, but these important improvements enable SSD to overcome these shortcomings to a certain extent.

B. Data solution

1) Data collection: Data collection is the process of gathering information from existing sources. In order to use the data we collect to develop practical artificial intelligence and machine learning solutions, it must be collected and stored in a way that makes sense for the problem at course. In this project, we collected the data set about the rubbish and wanted to make the camera could classify the kinds of rubbish with the data. Predictive models are only as good as the data from which they are built, so good data collection practices are crucial to developing high-performing models.

2) Data collection in this project: In this project, we need to divide the garbage into four categories: plastic bottles, cans, paper boxes and glass bottles. However, considering the small number of glass bottles, they are not usable, so the image data of plastic bottles, cans and paper boxes are finally needed.

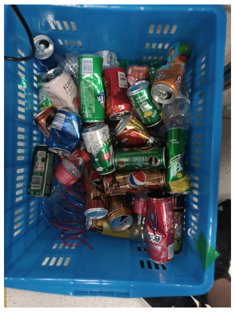


Fig. 5: Can

Fig. 6: Paper box

The program we use to collect the image data is provided by the course, which is ME336-2021Spring/deepclaw /utils/ImageDataCollection.py. This program collected the color, depth and infrared information of the rubbish by using the Intel RealSense camera. In the process of data collection, we randomly (including the space and orientation) placed the rubbish on the running convey belt. After collecting all the rubbish we have, we got the files with the image data.

3) Data augmentation: The quality of the taken picture is largely suffering corruptions from lighting conditions, weather, obstacles, motion distortions etc. In this condition,

Fig. 7: Bottle

using data augmentation to simulate the above distortion to expand the data set can effectively improve the recognition accuracy. Considering about the type of collecting data, we use Gaussian noise injection, horizontal blurred, horizontal flipping and vertical flipping to extend the data set. For example, we got this image for the color folder, and the augmentation results are showing below.

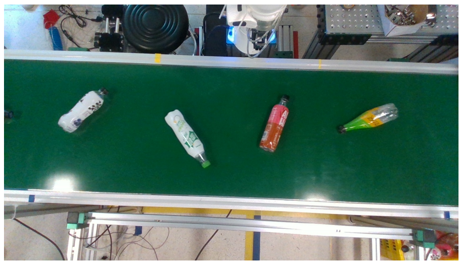


Fig. 8: Original image

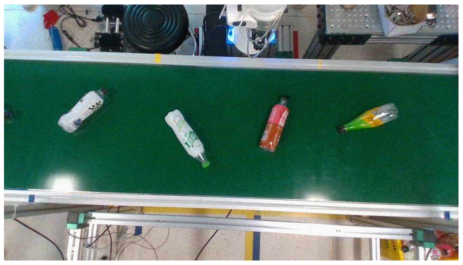


Fig. 9: Gaussian noise

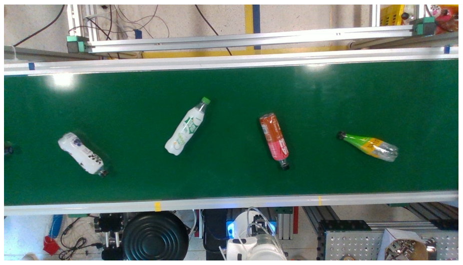


Fig. 10: Horizontal flipping

4) Data labeling: Data labeling is an essential part of data preparation, with simply tagging assigns labels to a set of raw data to make it easier to identify for predictive analysis ,and

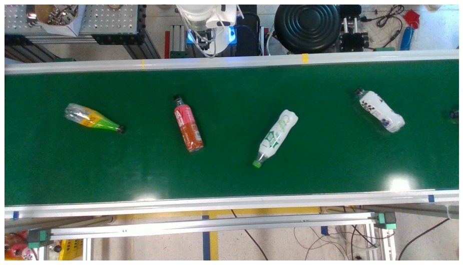


Fig. 11: Vertical flipping

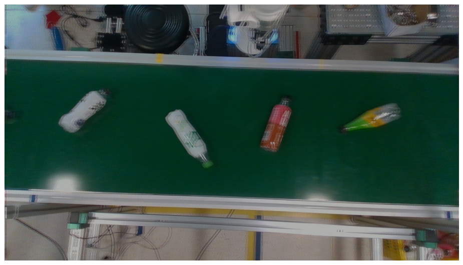


Fig. 12: Horizontal blurred

the label will show the kind of a image. In this project, we firstly want to use the automatic labeling provided by the course, but there were lots of problems occurring when we were trying to use the program. So, we decided to use the software LabelImg to label the data manually. The results of our labeling will be shown in the experiment result parts.

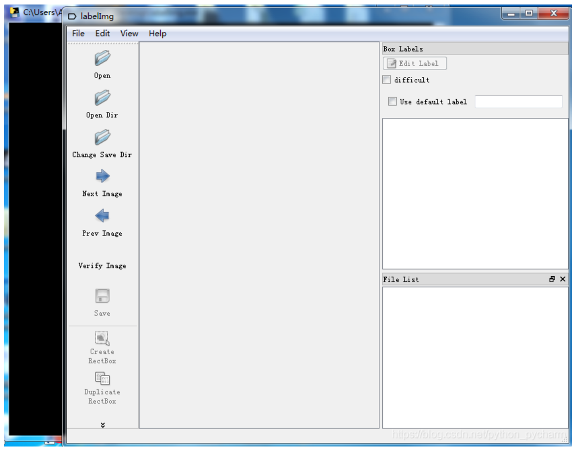


Fig. 13: The user interface of LabelImg

C. Software solution

1) Object detection: we will implemented object detectors on a depth camera using some popular pre-trained model such as YoloV3, SSD and so on. The code is written in C++ because the Intel realsense series are supported strongly by this language. Beside object detection task, The camera also give us the depth information of the object with quite high accuracy. There are three primary object detection methods that we will likely encounter: Faster R-CNNs (Girshick et al., 2015); You Only Look Once (YOLO) (Redmon and Farhadi, 2015); Single Shot Detectors (SSDs) (Liu et al., 2015). Faster R-CNNs are likely the most heard of method for object detection using deep learning; however, the technique can be difficult to understand (especially for beginners in deep learning), hard to implement, and challenging to train. Furthermore, even with the faster implementation R-CNNs (where the R stands for Region Proposal) the algorithm can be quite slow, on the order of 7 FPS. If we are looking for pure speed then we tend to use YOLO as this algorithm is much faster. The problem with YOLO is that it leaves much accuracy to be desired. SSDs, originally developed by Google, are a balance between the former two methods. The algorithm is more straightforward than Faster R-CNNs. We can also enjoy a fast FPS at 22-46 FPS depending on which variant of the network we use. SSDs also tend to be more accurate than YOLO. Because of the advantage of SSDs, we will use SSDs method to detect object.

2) Object grasping: The object grasping is to realize 2D grasping with manipulator. The 2D hand-eye calibration of the robotic arm and the camera is to obtain a perspective transformation matrix that can convert a point (u, v) in the camera's pixel space into a point on a plane in the base coordinate system of the robotic arm (x , y), at this time z is usually fixed, such as the height of a fixed horizontal desktop, (x, y) is the coordinate value of a certain point on the desktop relative to the base of the robot arm. We use the built-in function "cv2.getPerspectiveTransform(pts1, pts2)" provided in OpenCV to obtain the perspective transformation matrix M. The perspective transformation can be expressed by the following equation. Then we collect 4 sets of (u, v) and the corresponding (x, y), and get the perspective transformation matrix M by solving the equation. The color image obtained by the sensor and the visual recognition algorithm are used to calculate the grasping point, and the manipulator is controlled to reach the grasping point to complete grasping. In this experiment, the latest YOLO5 object detection algorithm was used to obtain the object Bounding Box, and the grabbing point was the center of the object Bounding Box.

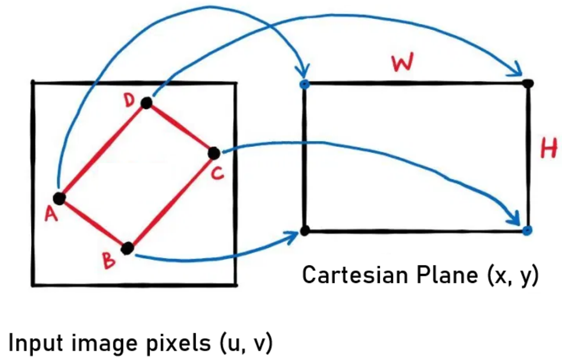


Fig. 14: Coordinate transformation

D. Design solution

III. Experiment Results

A. Design Experiment

The camera bracket is designed based on the structure of Realsense Camera D435. Adjust the camera and aluminum frame according to the desktop working area. The basic data of the camera is: the minimum depth distance is 10cm; Depth Z error is less than 2 percent and range is 2m; Depth image resolution 1280×720 (30fps); The field of view Angle is 86°×57°, equivalent to the camera's 24mm×40mm focal section.

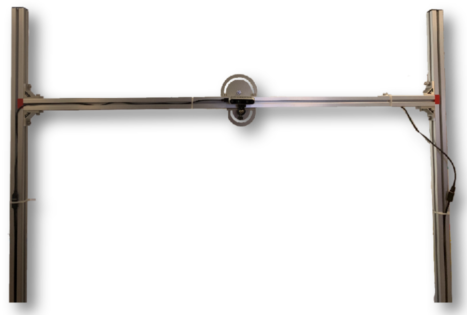


Fig. 15: Camera support

We designed three generations of models. The first generation model is an independent three-wheel bracket, which is easy to move and capture images by the camera. But the disadvantage is that the camera field of vision is too narrow, the field of vision is not broad enough, so it can not be used. The second generation is fixed on the side of the computer desk upright bracket, the advantage is a good vision, convenient disassembly; Disadvantage is easy to be encountered by the operator, the safety is poor.

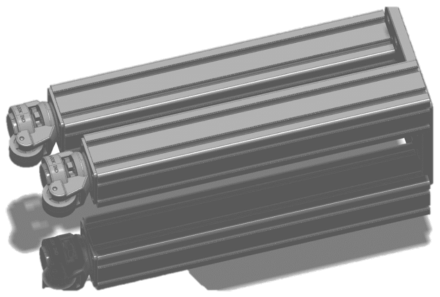


Fig. 16: Camera support1

Finally, we chose to directly fix the camera on the other side of the robot arm image capture system, which not only saves cost, is simple and beautiful, but also has good security, and the field of vision has been maximized.

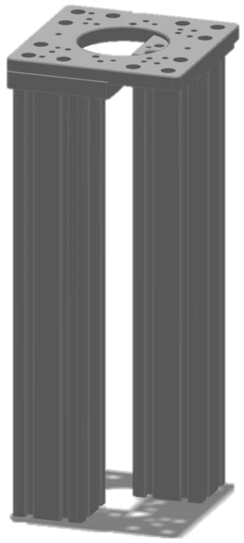


Fig. 17: Camera support2

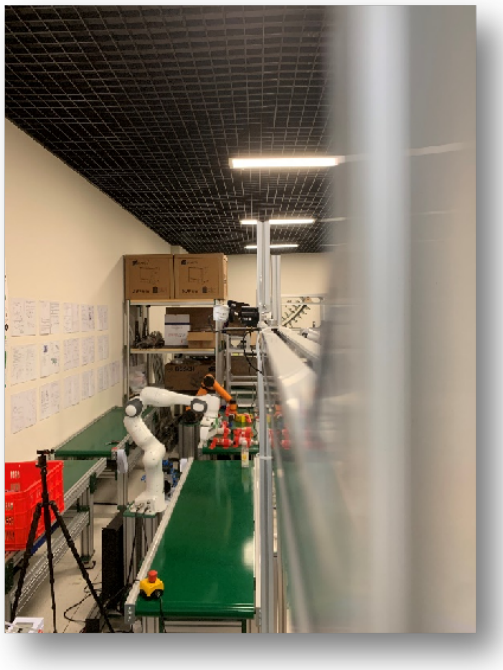


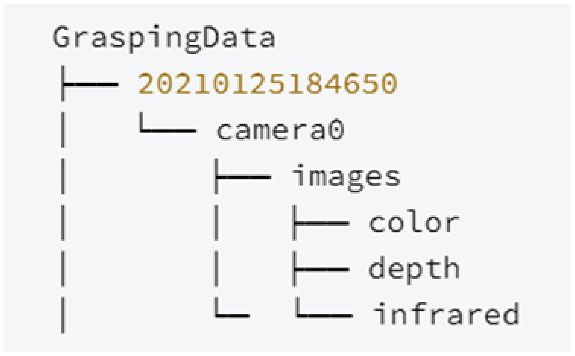
Fig. 18: Camera support3

B. Data Experiment

For a machine learning project that includes classification problems, data set is undoubtedly a very important part. The experimental part of data set in this project mainly includes:

1. Data collection: Collect the data of metal litter, plastic litter, and paper litter. The collection method is continuous photo collection, that is, place the litter on a moving conveyor belt one by one, and run the image data collection code so that the RealSense camera will record this process at a certain frame rate to get the data. The final result is each frame of image in the recording process. There are three

types of images recorded in this procedure, including: color image, depth image, and infrared image. Only color images are used in our project. The complete file structure of the data is shown in the figure 2 below.



```
GraspingData
├── 20210125184650
│   └── camera0
│       ├── images
│       │   ├── color
│       │   ├── depth
│       └── └── infrared
```

Fig. 19: Structure of collected data

2. Data labelling: Label the collected color image data. The labeling method can be divided into manual labeling method and automatic labeling method. Probably due to the problem of python virtual environment, the program will report an error when we run the automatic labeling code, so with the help of the LabelImg software, we manually label our collected data.

The experimental equipment used in this part includes: RealSense camera, conveyor belt, personal computer, metal litter (various styles of cans), plastic litter (various styles of plastic bottles), paper litter (various styles of cartons)

The experimental process of this part is as follows:

1. Data collection:

Before the data collection experiment starts, we need to prepare the objects to be collected and create a folder to store the data. For these three types of litter, we collect their data in three times, each time collecting one type of litter's data, so we put the same type of litter into a box, and place the box next to the beginning of the conveyor belt so that the rubbish can be placed on the conveyor belt in a convenient way, and then place an empty box at the end of the conveyor belt to collect the litter that fall from the conveyor belt after collecting the data. Then, we created three folders named "metal", "plastic" and "paper" under the specific directory location of the project folder to store the collected data.

Next, we open the "ImageDataCollection.py" code in the project file in Pycharm, and set the path in the code using the path of the previously created folder to store a certain type of litter data. Then, we turn on the conveyor belt by using the conveyor belt controller, set the conveyor belt speed at 40 percent (0.2m/s), and turn on the lighting in the fill light controller to adjust the brightness to the appropriate level. Finally, we run "ImageDataCollection.py" code, after the camera view appears on the monitor, we take out the litter from the box and place them on the conveyor belt one by one, with a certain distance between each litter, until all the litter in the box is placed, we end the process.

2. Data labelling:

As mentioned earlier, after the data collection process, we got three kinds of images, and in this step, we only use color images. First, we create a label folder in a specific location under the previously created folder for storing various types of litter data to store label files. Since LabelImg software provides function of exporting label file as different formats, including formats such as YOLO and PASCALVOC, as a result we correspondingly create two label folders "dabiaoyolo" and "dabiaopascal" to store both formats, for the convenience of algorithm and software engineers.

Then, we run the LabelImg software, open the folder of color images of a certain type of litter collected in the previous step, and mark the images in the folder: we open an image and manually draw a rectangle to select the litter part in the image, and then select the corresponding label, then we perform the same operation on all litter in this image, and after completing these steps, we export the label file of this image. Because the camera recording frame rate is very high, we don't need to label every image in the data folder, instead we just select a few images, which can totally cover every litter used. Figure 3 shows the process of labelling one image.

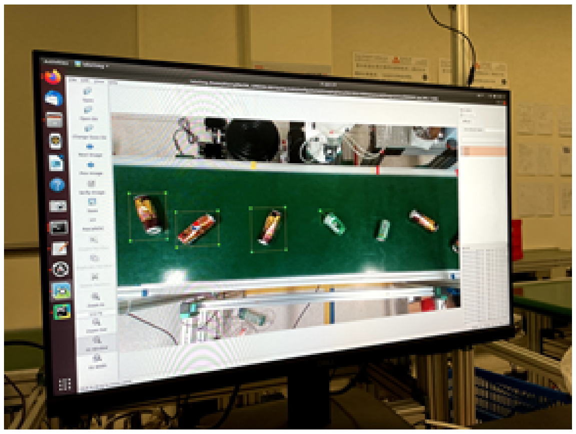


Fig. 20: Labelling collected color image

C. Software Experiment

In this experiment, a Realsense D435 camera was placed above the robotic arm. After the camera obtains the image and depth information. First, we use the human detection algorithm based on SSD to identify the person. By calculating the average depth value of the frame of the recognized person, we can get the depth distance between the D435 camera and the person. So we can get the distance between the human and the robotic arm. Then we need to grab the garbage and place it in the garbage basket. Here we use 2d picking method. Use the object detection algorithm based on YOLO, identify the garbage and get the coordinate value of the garbage. The movement trajectory of the robotic arm is planned by using the move-p instruction to realize the operation of the robotic arm to grab garbage and place it in the garbage basket.

Finally, according to the distance between the person and the D435 camera obtained before, the speed of the robotic

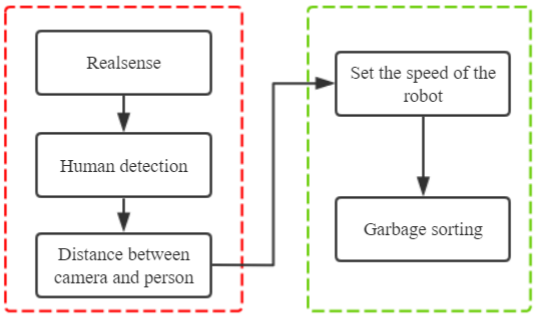

Realsense
Human detection
Distance between camera and person
Set the speed of the robot
Garbage sorting

Fig. 21: The main process

arm is adjusted in real time. After we get the distance of the person closest to the camera, we will make a judgment based on this distance, as shown in figure. There are three situations, the distance greater than 2.5m, the distance less than 1.5m, and the distance between 1.5m and 2.5m. We will set the speed of the robotic arm in stages according to these three situations. If this distance is greater than 2.5m, then this situation is considered a safe situation. When setting the speed, the speed of the robotic arm remains unchanged. If this distance is less than 1.5m, then this situation is considered a dangerous situation. When setting the speed later, the speed of the robotic arm will become very slow. The speed of the robotic arm is set to 20% of the initial speed. If the distance is less than 2.5m but greater than 1.5m. Then when setting the speed of the robot arm, the speed of the robot arm is 50% of the initial speed.

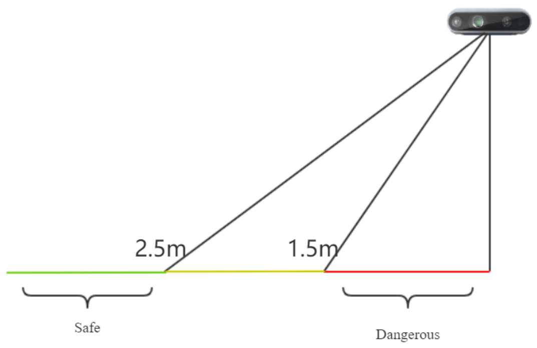


Fig. 22: Distance between human and camera

In the process of real-time setting of the robot arm speed, we used two methods. The first method is to recognize the person and set the speed before each move-p instruction of the robot arm. We use the human detection algorithm based on SSD, the recognition speed is fast, and the average time of processing a frame is 0.06s. Therefore, the control effect of this method is very smooth. The second method is to use multi-threading. Multi-threading is the ability of a central processing unit to provide multiple threads of execution concurrently. During the movement of the robotic arm, the speed of the robot arm is controlled in real time. At the same time, due to the characteristics of multi-threading, we can also get a visual window for observing the result of the human detection. And the final control result is also very smooth.

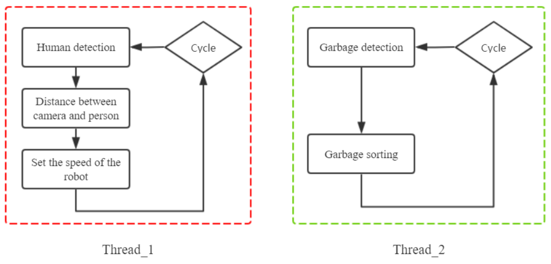
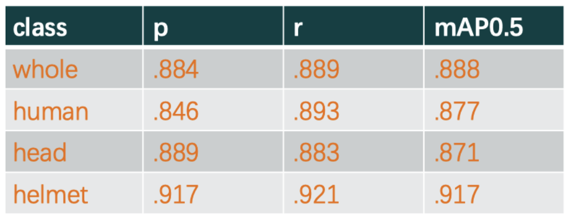


Fig. 23: Multi-thread process

D. Algorithm Experiment

In the experimental process, we used YOLOV5 algorithm for identification, and compared the effect with the SSD algorithm which was finally used. First, we determined the head, body and helmet as the recognition objects of the model. Since the original YOLOv5 had no interface, we wrote a programming interface to run through the main function's big loop. During the training, we used the Safety Wearing-Dataset, an official Dataset found on GitHub. YOLOv5 is divided into V5S, V5L, V5M and V5X. Through comparative experiments, we chose V5S with faster computing speed, whose recognition speed for a single image is about 0.03s. However, the recognition effect of V5S is poor for people who are far away from the camera, and objects beyond 2.5m may also be misidentified. The training results are as follows: After training, the mAP of



class	p	r	mAP0.5
whole	.884	.889	.888
human	.846	.893	.877
head	.889	.883	.871
helmet	.917	.921	.917

Fig. 24: The learning result

the overall model can reach 0.9. Since the program we wrote identifies heads, helmets and people in a series cycle, the final calculation time is the superposition of three recognition times, and the final calculation time is about 0.07-0.09s. At the same time, since YOLOV5 has poor recognition ability for people from 2.5 meters away, we finally try to use SSD algorithm for recognition, and only the whole person is identified. The time required for SSD algorithm to recognize a single image is about 0.04s-0.06s, which better meets the efficiency requirements of the program and ensures the accuracy of remote character recognition. At the same time, small objects will not appear the phenomenon of misidentification. Finally, the SSD algorithm was used to identify people within 4m with good results.

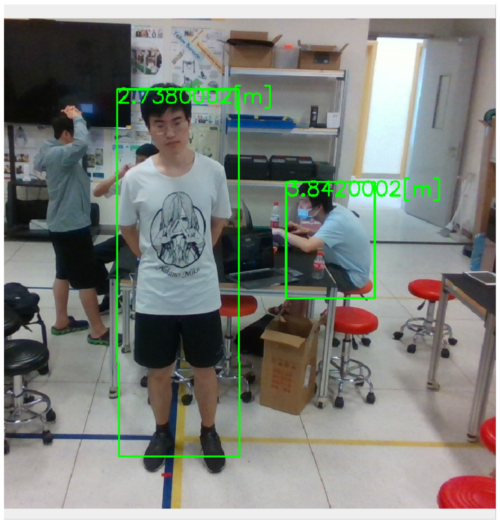
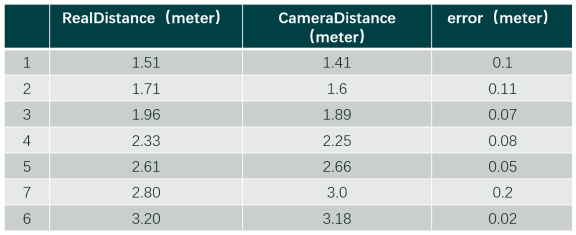


Fig. 25: The identification result

E. *System Experiment*

First, we tested the system's ability to recognize people. We measured 6 sets of data, namely the actual calculated distance and the camera display distance. It can be seen from the figure 21 below that the error of the human body recognition and ranging function is about 6 percent, which meets the requirements of system operation.
Then we tested the system as a whole, and the test video can be seen in the presentation of the report. During the test, the robotic arm realized the function of speed regulation according to the distance from the human body.



	RealDistance（meter）	CameraDistance（meter）	error（meter）
1	1.51	1.41	0.1
2	1.71	1.6	0.11
3	1.96	1.89	0.07
4	2.33	2.25	0.08
5	2.61	2.66	0.05
7	2.80	3.0	0.2
6	3.20	3.18	0.02

Fig. 26: The main process

IV. Summary

In general, our group realized the perception of the surrounding environment during the process of robotic arms grabbing garbage. Allows the robotic arm to adjust its running speed according to the surrounding environment. In this course project. We have completed the installation and debugging of the pneumatic gripper. The SSD and YOLO algorithms are used for human body recognition, and the two models are evaluated and compared. The collection of different types of garbage data has been completed, and the tasks of environment perception and garbage capture have been completed using multi-threading. However, our team still has certain shortcomings in its work. Because of the limited amount of data, we have not been able to classify the garbage; and there is blockage during the operation of the robot arm, and the phenomenon of jamming may occur during the operation.

Acknowledgment

Thanks to the teachers and TA for their help this semester

References

[1] Martin J. Rosenstrauch,,Tessa J. Pannen Jörg Krüger.(2018).Human robot collaboration - using kinect v2 for ISO/TS 15066 speed and separation monitoring. Procedia CIRP(),. doi:.
[2] Nikolaos Nikolakis,,Konstantinos Sipsas Sotiris Makris.(2018).A cyber-physical context-aware system for coordinating human-robot collaboration. Procedia CIRP(),. doi:.
[3] Roni-Jussi Halme,,Minna Lanz,,Joni Kämäräinen,... Antti Hietanen.(2018).Review of vision-based safety systems for human-robot collaboration. Procedia CIRP(),. doi:.
[4] Takeshi Morita,,Shunsuke Akashiba,,Chihiro Nishimoto,... Takahira Yamaguchi.(2018).A Practical Teacher-Robot Collaboration Lesson Application Based on PRINTEPS.. The Review of Socionetwork Strategies(1),. doi:.
[5] Petitti, Antonio,Di Paola, Donato,Milella, Annalisa,Lorusso, Adele,Colella, Roberto,Attolico, Giovanni Caccia, Massimo.(2016).A Network of Stationary Sensors and Mobile Robots for Distributed Ambient Intelligence. IEEE intelligent systems(6),. doi:.
[6] Bernard Schmidt Lihui Wang.(2014).Depth camera based collision avoidance via active robot control. Journal of Manufacturing Systems(4),. doi:.