Hardware/Software Design Description

For

Table Tennis Ball Collection Robot

Version 2.1 approved

Group 10

Date 10/05/2024

Revision History

Name	Date	Reason For Changes	Version
Generation of Elephant Mechanic Arm *MechArm*			
Equipment decision	2024-03-05	Initial version for MechArm	1.0
MerchArm assemble	2024-03-12	Preparation	1.1
MerchArm API test	2024-03-17	Preparation	1.2
Table tennis balls detection by MerchArm camera	2024-03-23	Foundation of collect table tennis balls	1.3
Claw to pick up balls	2024-04-02	Foundation of collect table tennis balls	1.4
Box (store balls) assemble	2024-04-16	Completed ball collection	1.4
Generation of DJI Robot			
DJI connection and test	2024-03-05	Initial version for DJI	1.0
Change source to DJI video stream	2024-03-12	Foundation of detect table tennis balls	1.1
Identify ping-pong balls and get coordinates	2024-03-23	Foundation of detect table tennis balls	1.2
Align the cart Angle with the ping pong ball	2024-03-29	Completed table tennis ball detection	1.3
DJI movement strategy	2024-04-07	Complete robot movement	1.4
Combination			
Test1	2024-04-23	Link MechArm and DJI	2.0
Test2	2024-04-26	Deal with the moving carton	2.1
Test3	2024-05-21	Deal with running lag on Nano	3.0

Table of Contents

1. INTRODUCTION

1.1. Purpose

This document describes the preliminary design of Table Tennis Ball Collection Robot project.

Our group plan to design a robot that can identify pin pang balls to collect them as well as detect the balls quality to classify them and put them to proper positions. To sum up, the table tennis ball collection robot will include the functions of ball identification, table tennis balls collection, quality inspection.

1.2. Scope

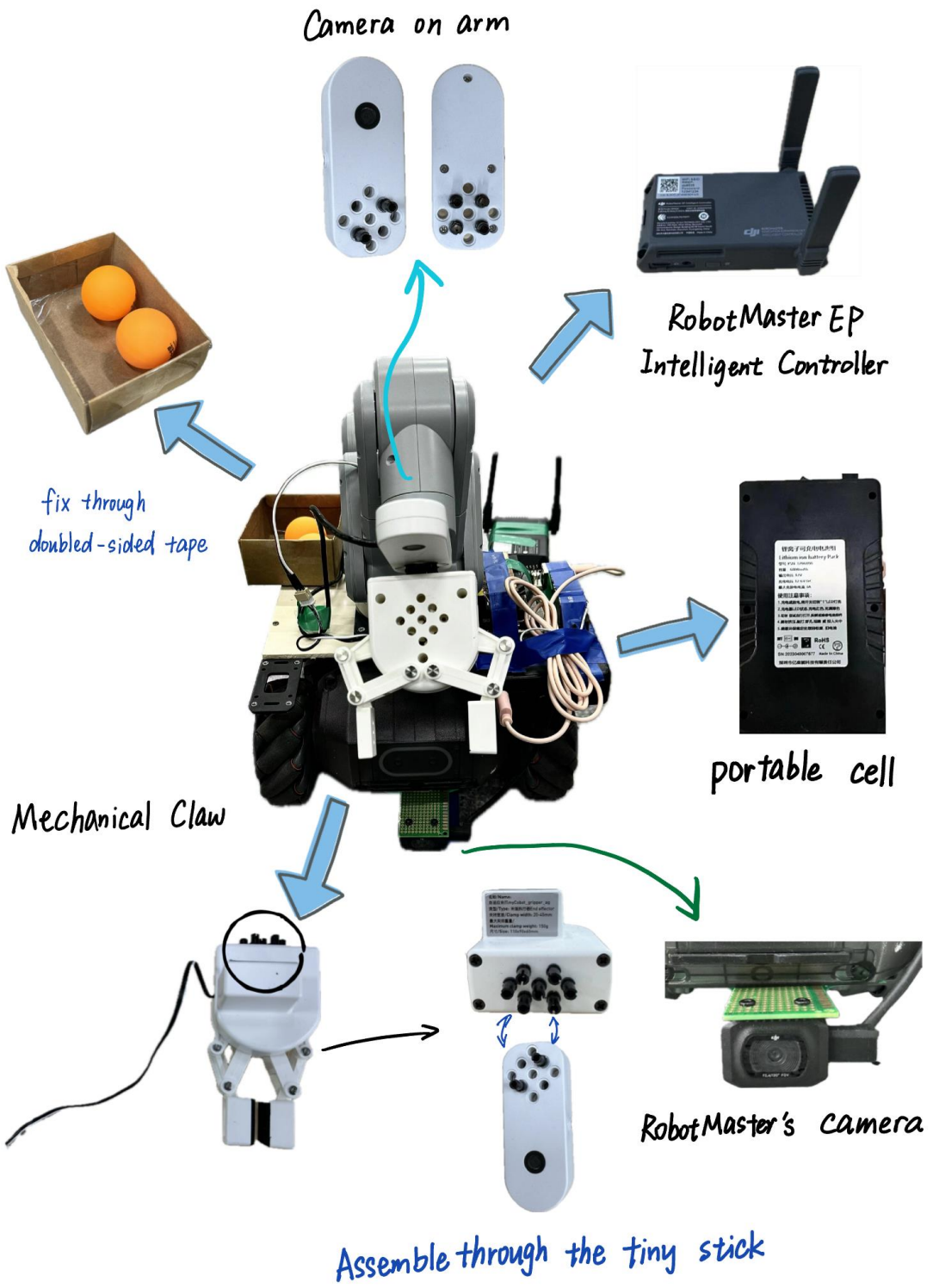
This document describes the range of our design application.

In today's rapidly developing economy, sports have become increasingly popular. As the national sport, table tennis is beloved by the masses. However, during play, the nature of table tennis causes the balls to scatter, requiring a significant amount of time to retrieve them, preventing players from fully enjoying the game. What’s worse, they may find the balls they pick up are damaged and useless.

Our production serves the table tennis players including the freshman who may need to collect table tennis balls more frequently and table tennis players who do not want to be interrupted by picking up balls.

2. HARDWARE DESIGN

2.1. Robot Setup




Elephant Robotics

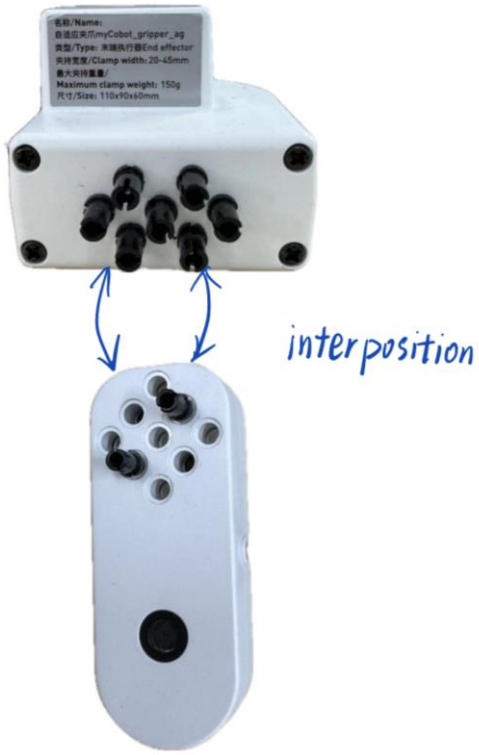
2.2. Camera Selection and setup

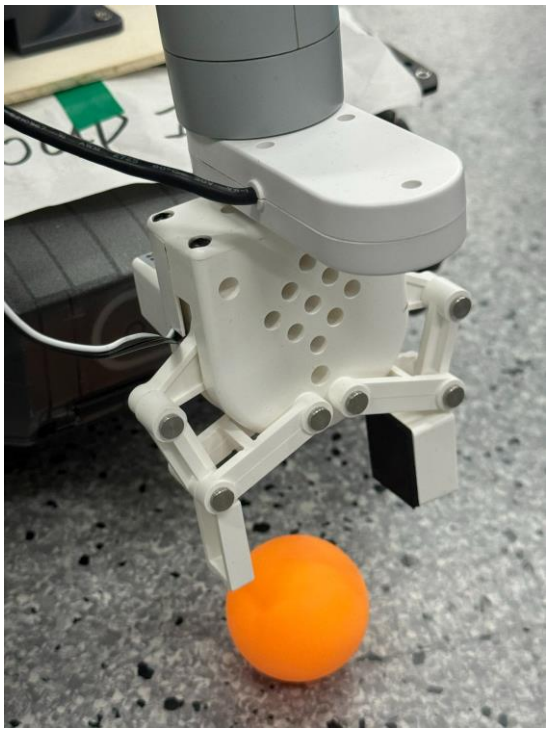
Camera selection:

Model: myCobot camera module v2.0

Material	ABS injection molding
Size	83*64*16
weight	50g
USB protocol	USB2.0 HS/FS
lens focal length	standard 1.7mm
field of view	about 60°
supported systems	Win7/8/10, Linux, MAC
operating enviroment	normal temperature and pressure
service life	two years
fixing method	Lego connectors
applicable equipment	ER myCobot 280 series/ER myPalletizer 260 series/ER mechArm 270 series

Camera setup:





DJI camera：

Material	**ABS injection molding**
Visual Angle	120 °
Maximum photo resolution	2560 x 1440
Video resolution	FHD: 1920×1080 30 pHD: 1280×720 30p
Maximum video bit rate	16 Mbps
Video format	MP4
Picture format	JPEG
Image sensor 1/4 inch CMOS	Effective pixels 5 million

2.3. Basic Computer Requirement

Operating system	*Linux-based Ubuntu Mate*
Processor	*Broadcom BCM2711*
RAM	*4GB*
Storage	*microSD card*
Graphics processor	*CPU：Broadcom BCM2711, 1.5 GHz,* *64-bit, 4cores,* *ARM Cortex-A72 Architecture,* *1MB shared L2 cache,* *Nvidia GTX 2080Ti*

3. SOFTWARE MODULES DESIGN

3.1. Architecture Overview

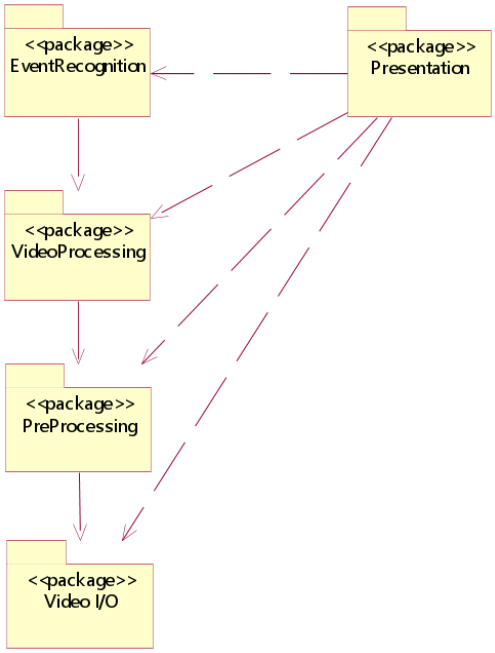


Figure 3.1 Package diagram

The system includes five packages: Video I/O package, Video Preprocessing package, Video Processing package, Event Recognition package and Presentation package, as shown in Figure 3.1.

3.1.1 Video I/O Package

The video is input into the system through two cameras: one camera in Elephant Robot Arm using OpenCV library to get the video input while another camera is the DJI camera adopting Robomaster method to get video input.

The output video after processing and logic could influence the robot movement, which is controlled by the Robomaster.

Input Package	Output Package
CV2、Robomaster	Robomaster

3.1.2 Yolo Target Detection

3.12. Video Processing methods are mainly gathered in Yolo frame. This project adopts Yolo v5s

framework. The library dependences are followed by the official documentation.

```
# YOLOv5 requirements
# Usage: pip install -r requirements.txt
# Base ----------------------------------------
gitpython>=3.1.30
matplotlib>=3.3
numpy>=1.23.5
opencv-python>=4.1.1
Pillow>=9.4.0
psutil  # system resources
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
thop>=0.1.1  # FLOPs computation
#torch>=1.8.0
# see https://pytorch.org/get-started/locally
torchvision>=0.9.0
tqdm>=4.64.0
ultralytics>=8.0.232
# protobuf<=3.20.1
# https://github.com/ultralytics/yolov5/issues/8012

# Logging -------------------------------------
# tensorboard>=2.4.1
# clearml>=1.2.0
# comet

# Plotting ------------------------------------
pandas>=1.1.4
seaborn>=0.11.0

# Export --------------------------------------
# coremltools>=6.0  # CoreML export
# onnx>=1.10.0  # ONNX export
# onnx-simplifier>=0.4.1  # ONNX simplifier
# nvidia-pyindex  # TensorRT export
# nvidia-tensorrt  # TensorRT export
# scikit-learn<=1.1.2  # CoreML quantization
# tensorflow>=2.4.0,<=2.13.1
# TF exports (-cpu, -aarch64, -macos)
# tensorflowjs>=3.9.0  # TF.js export
# openvino-dev>=2023.0  # OpenVINO export
```

```
# Deploy ------------------------------------------------
setuptools>=65.5.1 # Snyk vulnerability fix
# tritonclient[all]~=2.24.0

# Extras ------------------------------------------------
# ipython  # interactive notebook
# mss  # screenshots
# albumentations>=1.0.3
# pycocotools>=2.0.6  # COCO mAP
wheel>=0.38.0
# not directly required, pinned by Snyk to avoid a vulnerability
```

We list some important training details here. The dataset contains 300 images with txt format files for yolo training. The training process is conducted on Nvidia GTX 2080Ti with 100 epoches. The final mAP reached 0.988, indicating a good performance.

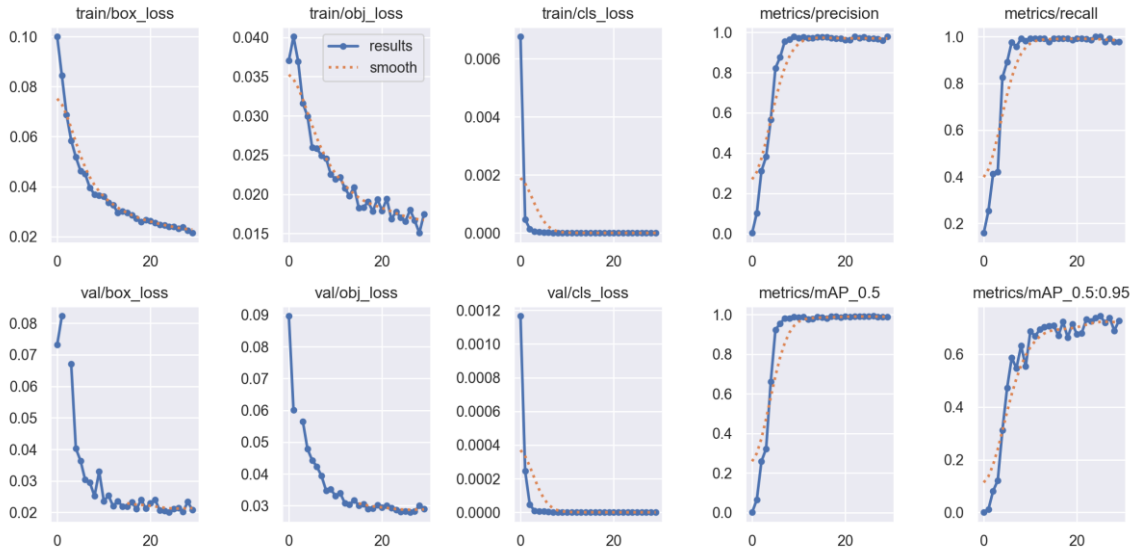


Figure 3.2: Training Results

3.1.3 Event Recognition Package

Recognition: We use the train-welled yolo model to detect the completed ball in the camera view. When we detect a ball, the recognition part goal is to return the target's coordination corresponding to the camera coordinate system. When a ball is picked up, the robot will rotate 90 degrees to find other balls until there are no balls after rotating 360 degrees.

Tracking: Since the coordination of the ball is passed through the recognition functions, the robot will firstly adjust its direction towards the ball. That means the robot will rotate a very small angle until the ball showing in the center of the screen. Particularly, it is an ideal condition, in real practice, we set an interval, meaning that if the direction is in the interval, there is no need to adjust again. Then the robot will move towards the ball. To calculate the moving distance, we utilize the labeling box square area to judge whether the current position is positive for picking up the ball. As a result, we set a threshold area, the robot will continue to move until the box area reach it.

Collection: When the gripper moves to the proper position, it tighten the paws to catch the table tennis ball then lifts and move towards the upper part of the ball collection box. Next, loosen the paws to put the ball into the box. Finally the robotic arm moves to the initial posture and get ready to next collection.

The total libraries for this part is listed below.

Package	Object	Function
CV2	/	HoughCircles
CV2	/	imwrite
Robomaster	Chassis	Move

Robomaster	Camera	start_video_stream
matplotlib	/	plot

3.1.4 Presentation Package

The packages need for this part is same with Video Processing packages since the interface is Yolo window.

3.2. Class Design

***** Description**

Class Name	yolov5
Inherited class	torch
Aggregated Classes	none
Associated Classes	cv2, numpt, scipy
Data Members	none
Member Functions	train, detect
Data Structures	convolution
Processing	target detection

Class Name	DJIRobot
Inherited class	Robomaster
Aggregated Classes	none
Associated Classes	none
Data Members	robot
Member Functions	init, forward, turn_left, turn_right, video_stream
Data Structures	none
Processing	basic movement and video operations of DJI robot

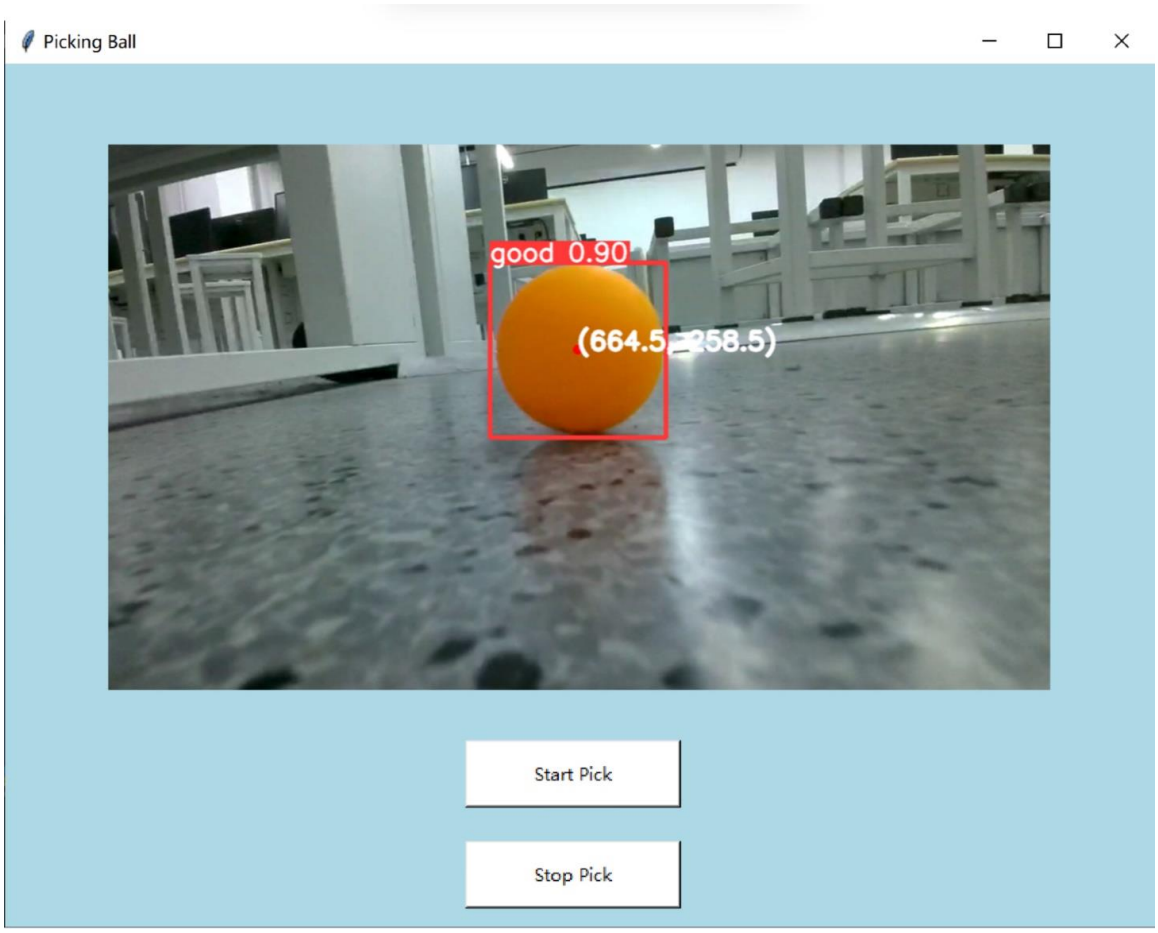
Class Name	armPick
Inherited class	pymycobot
Aggregated Classes	none
Associated Classes	cv2, numpy
Data Members	mc, angles
Member Functions	arm_adjust_total
Data Structures	queue
Processing	basic movement and video operations of machinical arm, table tennis ball recognization

4. USER INTERFACE DESIGN

4.1. Main Interface

It is really easy for user to make the table tennis ball robot to help them to collect and detect balls. Push the button then the robot will start to work automatically.

- Using WiFi mode to connect DJI to computer.
- Open python and run detect.py file.
- Press Start Pick to start collecting.
- Press Stop Pick to stop collecting.



Two devices are connected through Socket communication library in python. All the codes are integrated in Jetson Nano platform. To transmit data, Nano, DJI robot and elephant mechanical arm connect to same wifi. After that, we could run the program in nano to begin all the project.

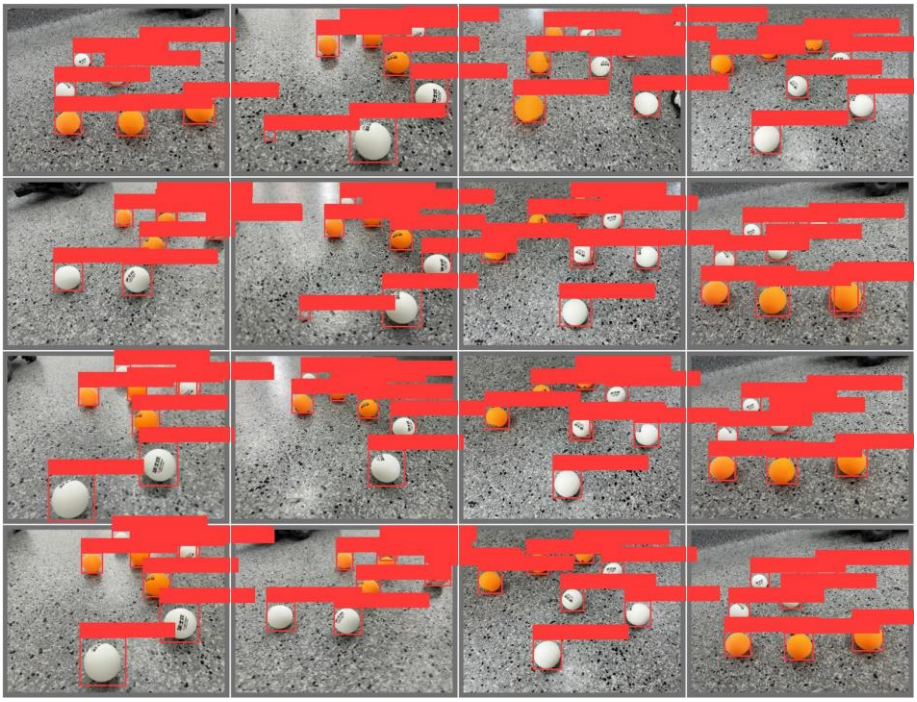
```
import argparse
import csv
import os
import platform
import sys
from botMove import DJRobot
from pathlib import Path
import torch
import socket
center_x, center_y = 0, 0
```

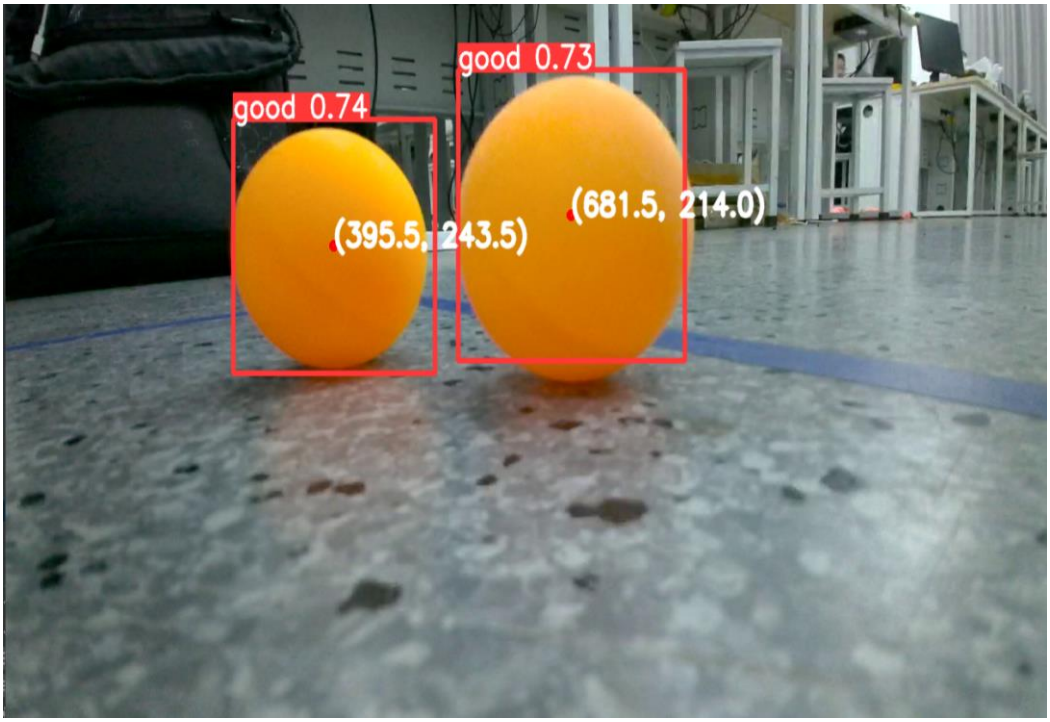
4.2. Table Tennis Ball Detection Window

Suppose user has installed the above environments and dependencies. Before the user run the main program, make sure the yolo detect parameters of source are sent the video stream address of mobile camera which is a constant address for DJI camera.

Alter the python interpreter to the environment that installed the requiring environments.

After configuring all the necessary dependencies, the yolo detect results could shown in the screen. The target will be labeled in red box and as well as the coordination printed in the screen.




good 0.74
good 0.73
(395.5, 243.5)
(681.5, 214.0)