README

Step 1: Set up Python Environment for proper Dependencies

1. Install MiniConda with Python 3.8
2. Create new conda environment (with Python 3.8)
   Tutorial: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html
   Alex laptop environment name: GRIP_env

Install following packages in your environment:

1. Opencv-contrib
2. Torch
3. Pandas
4. pyyaml
5. pillow
6. tdqm
7. matplotlib
8. seaborn
9. Set port correctly
10. Requests

Step 2: Preliminary set-up

1. Unzip Archive.zip
2. Go to FullVisualServoingUR5-V3.2.py
3. Set up path to weights file correctly. The weight that can achieve best performance is best.pt in current (master) folder.

4. Set up IMAGE_CENTER
   For endoscope of 1280*720, it is [640, 360]
5. Find camera parameters (Camera Matrix and Distortion Factor) for your camera
   How to find camera parameters:
   a. Print a chessboard
   b. Measure its square size
   c. Take some pictures (>30)
   d. In MyCameraCalibration.py, put in (1) inner corners (2) size of chessboard square (3) Folder containing all images of chessboard
   e. Run MyCameraCalibration.py to get camera parameters

```
YOLO_WEIGHT_FILE = "./yolov5/runs/train/exp13/weights/best.pt"
cameraMatrix = np.array([[927.31957258, 0,667.19142084],[0,922.20248778,335.69393703],[0,0,1]])
dist = np.array([[-0.17574952,0.65288341, -0.00300312,  0.00724758, -0.95447869]])
IMAGE_CENTER = np.array([640, 360])
J_PRIME = np.eye(2)
MODEL = None
```

Step 3: Test communication with Arduino

1. Change port number in SerialCommWithArduino.py

```
# Importing Libraries
import serial
import time

arduino = serial.Serial(port='/dev/cu.usbmodem1101', baudrate=115200, timeout=.1)
```

Here, port number is '/dev/cu.usbmodem1101'. Please carefully check your Arduino app and put port number there.

2. Upload CommWithPython1.ino to Arduino
3. Test grip() and ungrip() functions to see whether the gripper successfully communicates with your computer

Step 4: Run FullVisualServoingUR5-V3.2.py

python FullVisualServoingUR5-V3.2.py

This program has 4 features:

1. Visual-servoing (first calculate Jacobian and its inverse, then use it to center the blackberry in camera view)
2. Detection of multiple artificial blackberries to return their centroid and bounding boxes
3. Go for blackberries one by one (from leftmost one to rightmost one)
4. Perform closed-loop control (re-centering after each movement in z-axis)

Note:

best.pt in current folder works well for both artificial blackberries and real blackberries, even for those not seen before.

Sample images from Maybury farm visit:

Ground-truth labels:

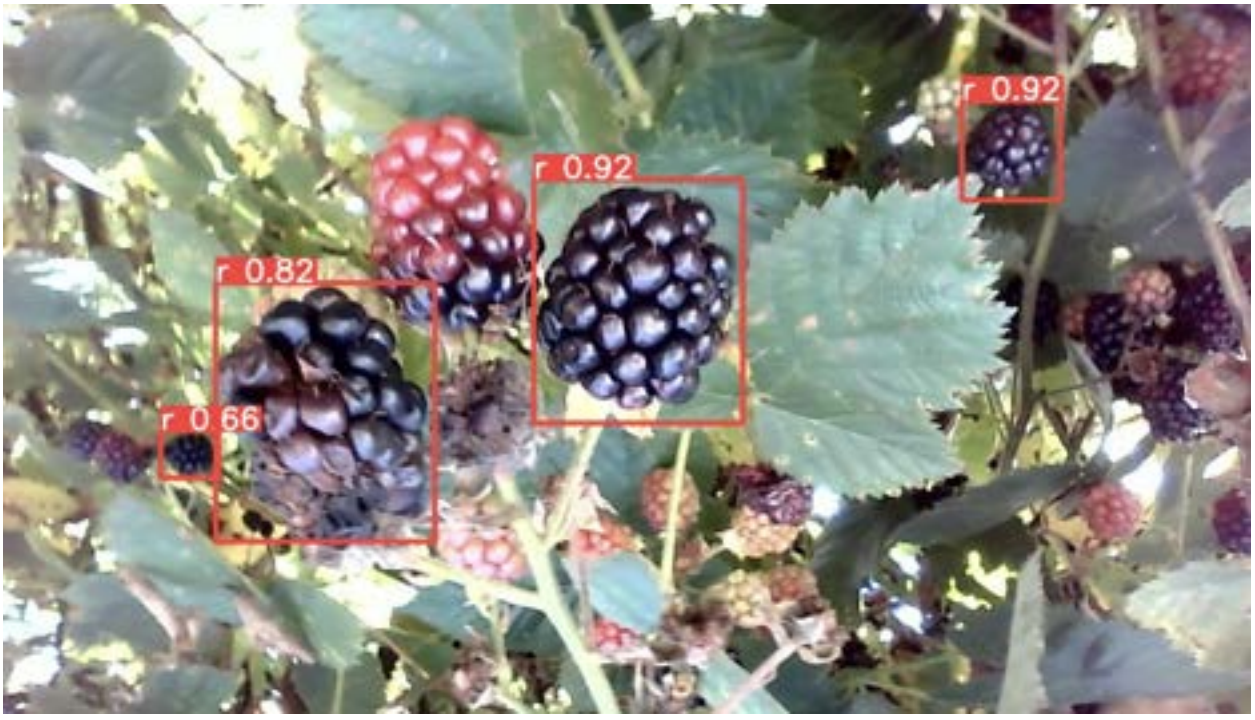
YOLO model labels:

Mean Precision: 0.824

Mean Recall: 0.645

More YOLO-labeled images.

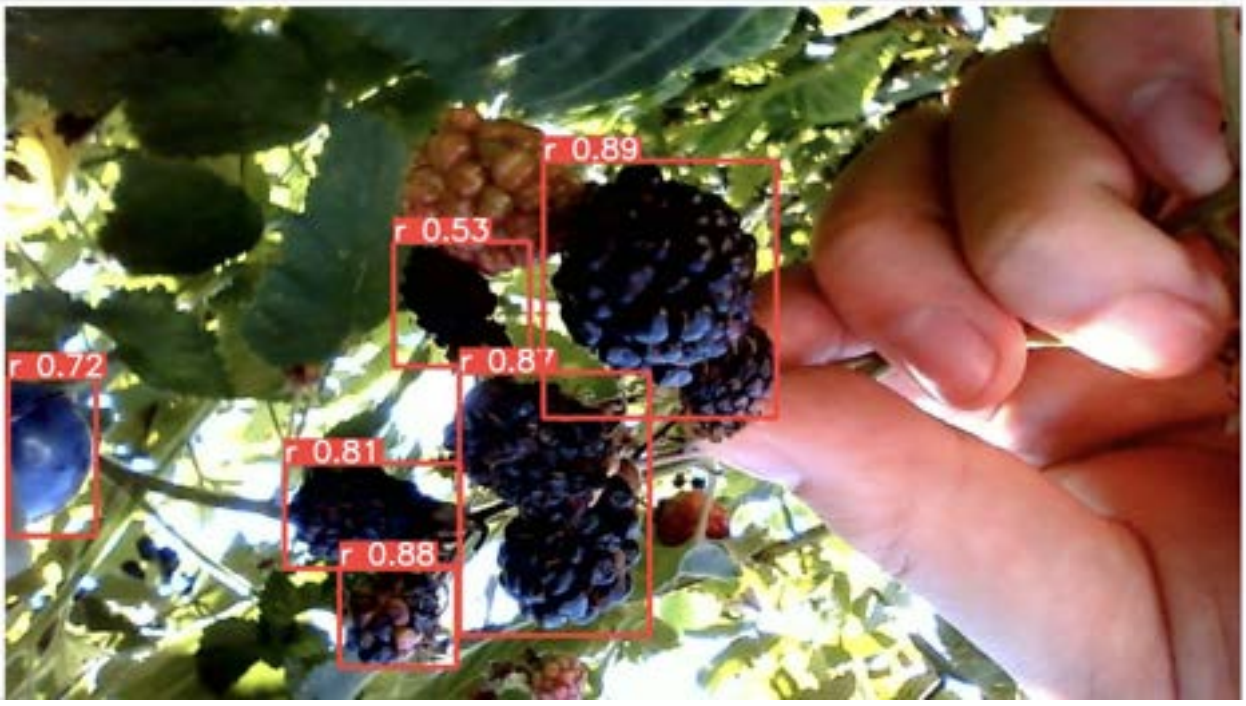

0.73
0.58
0.91
0.92
0.34
0.75

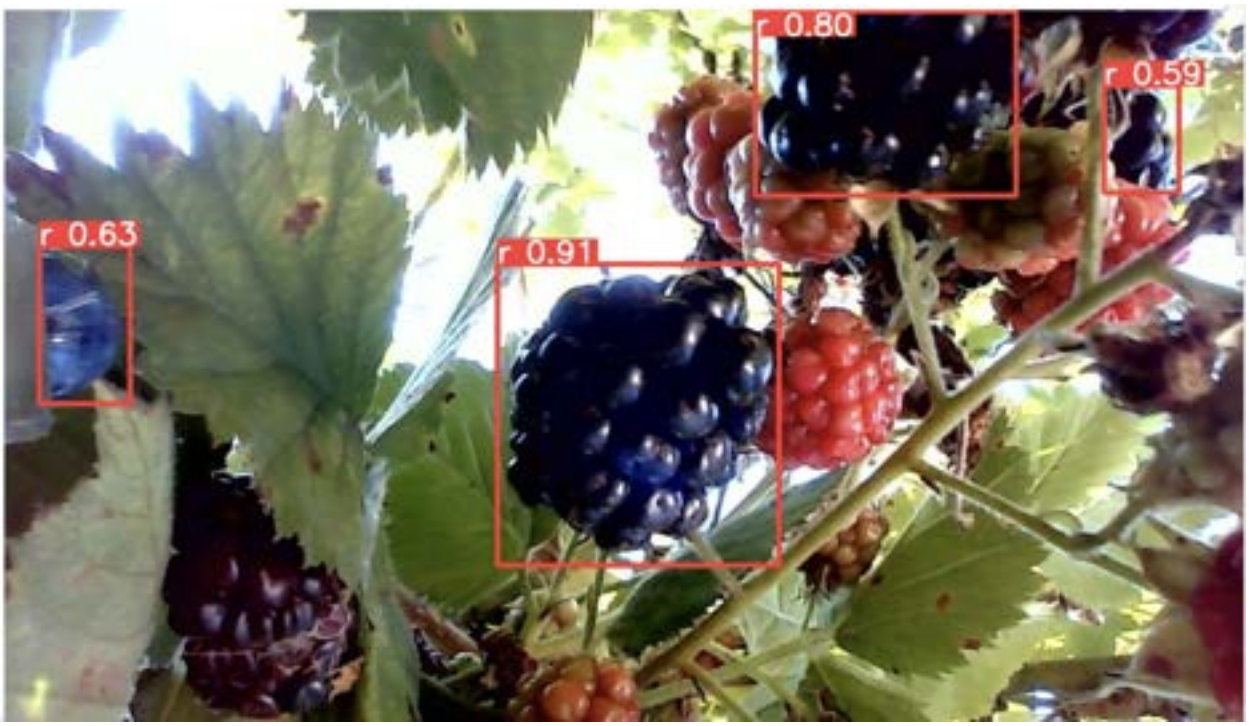

0.92
0.92
0.82
0.66


0.90
0.36
0.93


0.84
0.46
0.49


r 0.89
r 0.68
r 0.87


r 0.89
r 0.53
r 0.72
r 0.81
r 0.88


0.80
0.59
0.63
0.91