Biped humanoid robot combined with computer vision technology - For real-time object detection based on YOLOv3

劉力元 Li-Yuan Liu　　徐靖憲 Ching-Hsien Hsu

Instructor: 林柏江教授 Prof. Po-Chiang Lin

Abstract

This goal of this research project is combining with computer vision and humanoid robot that can fulfill particular task automatically. In this way, there will no longer relies on people to control. In this research project will need Nvidia Jetson Nano, Webcam, Siegfried robot and will need YOLO(You Only Look Once) or other object detection technique. We will compare different version and efficiency. In communication technology, we will use PySerial module to process Serial communication. The PySerial will send our instruction to Arduino which is control our humanoid robot. PySerial also control the servo motors so that the robot can react to our instruction and finish the task and movement.

Keywords: Object detection, autonomous robot

Index

1. Motivation and Purpose

According to the developing progress of robot's software and hardware, the robot field become more and more popular. Many researchers and research institutes study how to make our life more convenient with robot. In the background that GPU becomes more and more powerful, robot combines with deep learning and Artificial Intelligence more often to be used. Different to traditional way which is human controlling the robot, now the new method is using fully automatically way to control robot. For example, using rescue robot can decrease the death and can efficiently finish the rescue task. Thus, the goal of our project can combine deep learning with the robot and can use this robot in the real life.

2. Setting the original environment

2.1 Introduction to Robot Platform Environment

This project use the environment bellow：

(1) Dataset's label : Windows 10
(2) Training: i7-8700 + Titan x, Ubuntu 16.04
(3) Image Recognition: Jetson Nano
(4) Ubuntu 16.04

(4) Robot: Arduino Micro, Arduino 1.8.10

2.2 Jetson Nano's installation and setting

First step uses 16G SD card and downloads the image data[1] from the nvidia's website. In order to keep using Jetson nano with the insufficient memory, we need to install swapfile, which can add swap space(like virtual memory in windows) so that we can use more memory space. This step need to connect the monitor.

Fig. 1 The image of Jetson Nano

2. 3 VNC connection

In general, we will let Nvidia Jetson Nano connect to monitor, keyboard and mouse to control it. But if we want to go outside and we don't have monitor, keyboard and mouse. In this case, we will need to use remote connection by VNC. VNC connection needs the Jetson Nano and computer connected in the same internet domain. Use computer(Host) to control the Jetson Nano(Guest). Below is step to create the VNC connection.

(1) In windows local computer install VNC viewer as remote control's platform.
(2) In Jetson Nano install vino VNC and open dconf-editor close the encrypted connections.
(3) Setting the user name and password and open the auto login.
(4) Use initial startup program and set open the Vino VNC when the power on.

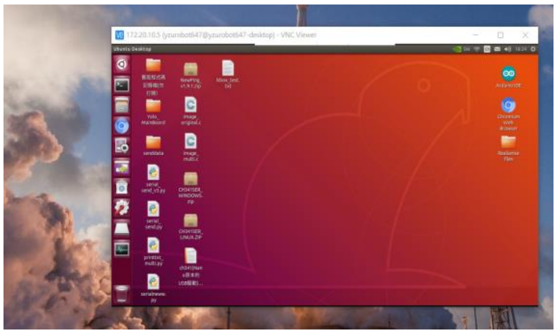


Fig. 2 Open VNC connection on Windows local computer

3. YOLO (You only look once) object detection model

3.1 YOLO(You Only Look Once) object detection module

YOLO series [2][3] are widely used object detection technique in recent years. YOLO's principle is to select the area depend on bounding box's information and select feature to classify the object. This method is more faster and accurate than others method like SSD[4], RetinaNet[5] and R-FCN[6]. According to the YOLO's official website, download the darknet installation package. Then we need to activate the OpenCV, CUDA and cuDNN. Finally, building the darknet and finishing the installation. Because Jetson Nano already has OpenCV, CUDA and cuDNN, we don't need to install them.

Next, we need to collect a lot of images as the training data. In this step, we use video to image transform tool every 1 second. And then, we need to prepare the training data. We use LabelImg to label the images, and the label output data type is xml. YOLO need .txt to train the data, so we use darknet file's voc_label.py to covert xml to txt. Use voc_label.py also create the txt file contain the training data's path and name. Next step we need to create the obj.data which contains the training path, weight-saved path and revise classes depending on our training data.

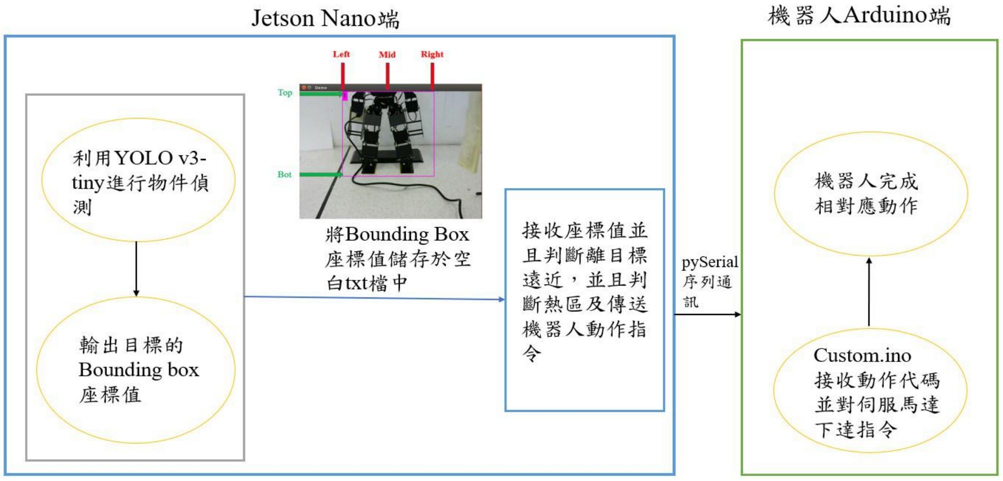
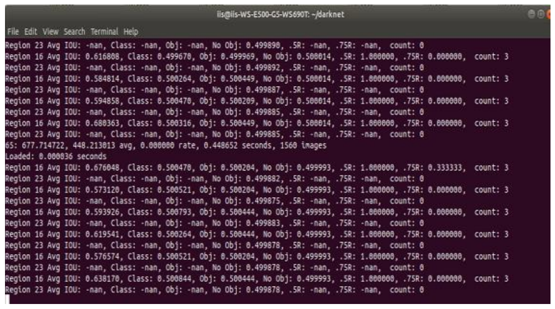

Jetson Nano端
機器人Arduino端
Left
Mid
Right
Top
Bot
利用YOLO v3-tiny進行物件偵測
輸出目標的Bounding box座標值
將Bounding Box座標值儲存於空白txt檔中
接收座標值並且判斷離目標遠近，並且判斷熱區及傳送機器人動作指令
pySerial序列通訊
機器人完成相對應動作
Custom.ino接收動作代碼並對伺服馬達下達指令

Fig.3 Robot’s operation process

When we finish these step, we need to revise the classes and filter’s value in yolov3-tiny.cfg data depend on our training data classes. After we finish that we can start training our data. We have 2 classes including training data 1954 and testing data 485.



```
iis@iis-WS-E500-G5-WS690T: ~/darknet
File  Edit  View  Search  Terminal  Help
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499890, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.616808, Class: 0.499670, Obj: 0.499969, No Obj: 0.500014, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499892, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.584814, Class: 0.500264, Obj: 0.500449, No Obj: 0.500014, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499887, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.594858, Class: 0.500470, Obj: 0.500209, No Obj: 0.500014, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499885, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.680363, Class: 0.500316, Obj: 0.500449, No Obj: 0.500014, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499885, .5R: -nan, .75R: -nan,  count: 0
65: 677.714722, 448.213013 avg, 0.000000 rate, 0.448652 seconds, 1560 images
Loaded: 0.000036 seconds
Region 16 Avg IOU: 0.676048, Class: 0.500470, Obj: 0.500204, No Obj: 0.499993, .5R: 1.000000, .75R: 0.333333,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499882, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.573120, Class: 0.500521, Obj: 0.500204, No Obj: 0.499993, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499875, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.593926, Class: 0.500793, Obj: 0.500444, No Obj: 0.499993, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499883, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.619541, Class: 0.500264, Obj: 0.500444, No Obj: 0.499993, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499878, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.576574, Class: 0.500521, Obj: 0.500204, No Obj: 0.499993, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499878, .5R: -nan, .75R: -nan,  count: 0
Region 16 Avg IOU: 0.638170, Class: 0.500844, Obj: 0.500444, No Obj: 0.499993, .5R: 1.000000, .75R: 0.000000,  count: 3
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499878, .5R: -nan, .75R: -nan,  count: 0
```

Fig. 4 Model training

3.2. Use webcam to process the object detection

When we finish the training of yolov3, we will get the weight data. We select the highest accurate data to process the object detection. Below images is the training information of yolov3. You can find that yolov3 start decreasing when the batches in 600.

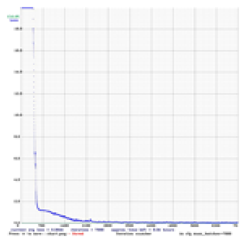


Fig. 5 The loss curve of YOLOv3

Next step, we connect webcam to the Jetson Nano and run the code on terminal. This code will detect the object showed on the terminal, also will show the fps and object’s accuracy on the terminal. Depend on the testing result used the yolov3-tiny. Compare to yolov4-tiny will found the yolov4-tiny more easier interference by background.

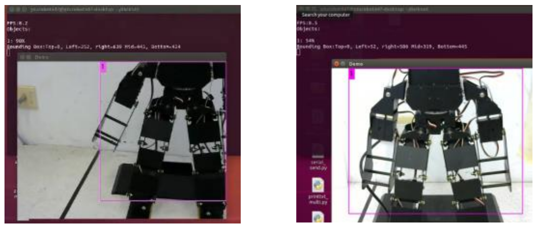
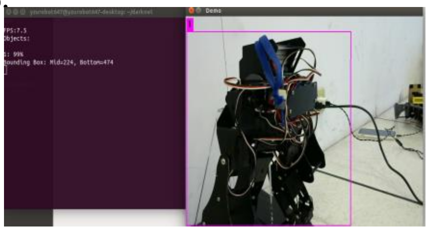


Fig. 6 YOLOv3-tiny model(left)、YOLOv4-tiny model (right)

3.3 Training result

We separately test the accuracy from different angle and distance. In the different angle, we test from front side and back side, front side and back side with 45 degree, front side and back side with 90 degree . In the different distance, we test from 42 cm, 26 cm and 12 cm separately. The accuracy basically reach 90%.



```
FPS:7.5
Objects:

1: 99%
Bounding Box: Mid=224, Bottom=474
```

Fig.7 YOLOv3-tiny backside 45 degree (99% accuracy)

In the same time, we randomly picked 50 test data and got the 50 data's confident value. The yolo v3 module's average test confidence value is 85.94.

4 Data transmission and execution between Jetson Nano and the robot

4.1 Hardware connection

In the power supply part for Jetson Nano. We uses power bank for power supply. The power bank model is ASUS ZenPower 3.75V 10050mAh. For the power supply for the robot, we use 1300mAh LiPo battery. For the hardware connection part, first we connect power supply with Jetson Nano, then use Logitech C270 webcam for detection. Moreover, we make a plastic box, which will install on the head of the robot, and put the Jetson Nano inside the box. Second, we connect Arduino to Jetson Nano which control the servo motor. The following is the connection diagram.

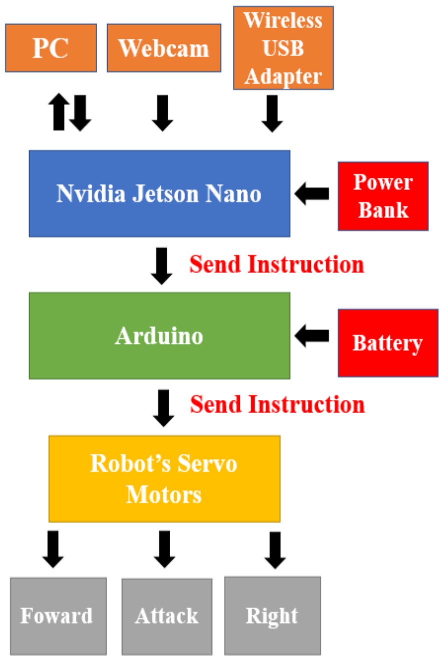


Fig 8. Hardware connection diagram.

4.2 Position coordinate processing of detected objects

When we using YOLO to run the object detection , we are not only need to find the category of the objects that we want to detect, but also have to analyze and deal with the object position coordinate. In the darknet version, we modify the draw_detection function which is in the darknet/src/image.c. First, we need create a new empty text file in darknet/src/ which is named as bbox.txt. The purpose is to modify the original output coordinate value of the detected object in the function of draw_detection in image.c. It will save the value we want, such as the class and the middle coordinate of the bounding box as well as the bottom value of the bounding box. Use fprint to write in the bbox.txt.

We need to define the area which is robot can hit the object. Therefore, we did some test and we found out the bounding box bottom value became bigger when the Siegfried robot close to the object. Though the many test, we set the bounding box bottom value to 470 which the Siegfried robot will attack the target.

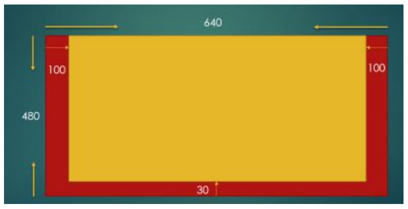


Fig 9 . The bounding box attack area's value(bigger than yellow area)

4.3 Pyserial Serial communication

Using Micro-usb to connect Jetson Nano and the robot. We use Python library- pyserial module to do serial communication. After finish the connect, the Jetson Nano can send the command to robot.

The module named “serial” automatically selects the appropriate backend, and complete the communication.

4.4 The detailed process of run.py

The following is the operation flow of run.py

1. First, open the terminal. Using YOLO command to open the Webcam and detect the object.

2. Second, set the connection port, and turn on the power switch of the robot.
3. Then run run.py in Terminal and enter the object number that we want to be detected. For example, input 0.

4. When the object is detected, the type of the input object will be temporarily stored in the cls variable in bbox.txt for each frame. If the specified object has not been detected, the "Find Object Command" will be executed. We design the algorithm that the robot will keep turning right until the specified object is detected.

5. If the value in bbox.txt file is being read; the program will keep searching the same class to verify the class is belongs to the same class for the following frame. Next step, it will send the command to Arduino, Corresponding to mid coordinate value and bot coordinate value, so the Arduino can control the servo motor and do corresponding actions, such as moving forward or turning right.

6. When the object's Mid coordinate value and Bot coordinate value reach to the offset value, we designed the action that robot will hit the object and end the program. The mission will end.

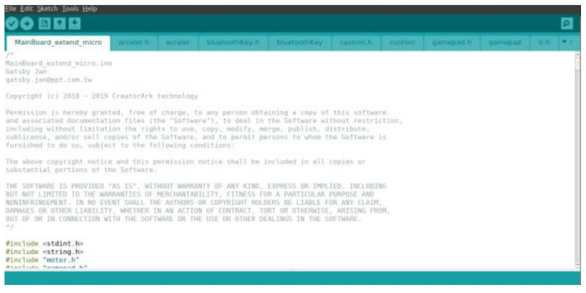
Algorithms 1: Robot operation algorithms

```
Input: Selected object class $SC$, Object bounding box $B$, total frame $T$, bounding box $B$, referenced mid range coordinate $C1$ $(C11, C12)$, referenced bot coordinate $C2$
Output: Robot right movement $R$, robot forward movement $F$, robot punch $P$
Activate robot
Input $SC$ on the terminal
for $f = 1,2\ldots,T$ do
    if $SC$ not in $f$ then
        $R = R + 1$;
    else
        $B_{sc,f} \leftarrow \{(\text{mid1}, \text{bot1})\}$
        if $Bsc,f[0]$ not in the range of($C1$) then
            $R = R + 1$;
        else if $Bsc,f[0]$ in the range of ($C1$) then
            $F = F + 1$ ;
        else if $Bsc,f[1] >= C2$ then
            return $P$;
        end
    end
end
Deactivate Robot
```

Above is the robot’s operation processing.

5. Robot settings and motion compilation

This capstone uses the Genuino Micro version of Arduino. First, select Genuino Micro in the Tools/Ports option, and import the custom.ino code of the of the Siegfried robot, and follow the original settings of the Siegfried robot. Modify the custom.ino code according to the custom output coordinate values from the bbox.txt. Moreover, use the motion compiler program provided by the original Siegfried robot to complete the motion compilation according to our needs.



```
/*
MainBoard_extend_micro.ino
Gatsby Jan
gatsby.jan@ppt.com.tw

Copyright (c) 2018 - 2019 CreatorArk technology

Permission is hereby granted, free of charge, to any person obtaining a copy of this software
and associated documentation files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*/

#include <stdint.h>
#include <string.h>
#include "motor.h"
```

Fig.10 Import the custom.ino code of the robot

6. Result

Suppose we want to find object 1, we can place multiple objects in front of the robot, such as object 1 in the figure below to check whether it will misjudge or not. The robot verifying the object in front of it is not object 1. In this case, it turns to the right and continues to look for target. After find the target, it will slowly rotate until the object is in the center of the screen, then the robot will move forward and approach the target, eventually knock down the object with a fist. Finish identification.

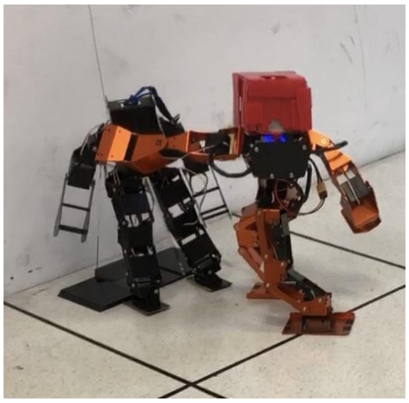


Fig. 11 Detect the specified object and knock down the object

7. Conclusions

In this capstone, we learned how to use the current SOTA object detection architecture YOLOv3. Also, we manually optimize the network architecture. It is extremely difficult for robots to detect the small targets from long distances, and there is still have a lot improvement such as accurately identify object and locate them within long distance and very close distance. At present, we continue to learn latest related work, and to study more difficult tasks. To identify all universal object in supervised learning method is still a challenging work. I hope that in the future, we can keep doing research on how to make the robot more faster to. detect the object as well as how to make the detection in real-time

8.Group member's work distribution

1. 劉 力 元 (1063706)object detection, model train, combine software and hardware, hardware setting, paper writing. (50%)

2. 徐 靖 憲 (1063740) object detection, model train, combine software and hardware, hardware setting, paper writing. (50%)

9.Budget

Name	**price**	**unit**	**Total**	**Use for**
Ethylene Viny Acetate	3	13	39	Assembly Robot's head
PP board	50	1	50	Used in Robot's head
Mi 10000mAh power bank	645	1	645	Supply power to Jetson Nano
Total			734	

10. Reference

[1] https://developer.nvidia.com/embedded/downloads

[2] Joseph Redmon and Ali Farhadi. Yolov3:An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[3] YOLOv4: Optimal Speed and Accuracy of Object Detection Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao

[4] W.Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed.SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015. 5, 6

[5] Focal Loss for Dense Object Detection Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár Mon, 7 Aug 2017

[6]R-FCN: Object Detection via Region-based Fully Convolutional Networks
Jifeng Dai, Yi Li, Kaiming He, Jian Sun 20 May 2016

[7] https://pjreddie.com/media/files/yolov3. Weights

[8]https://pjreddie.com/media/files/yolov3. weIghts

[9] https://swf.com.tw/?p=1188

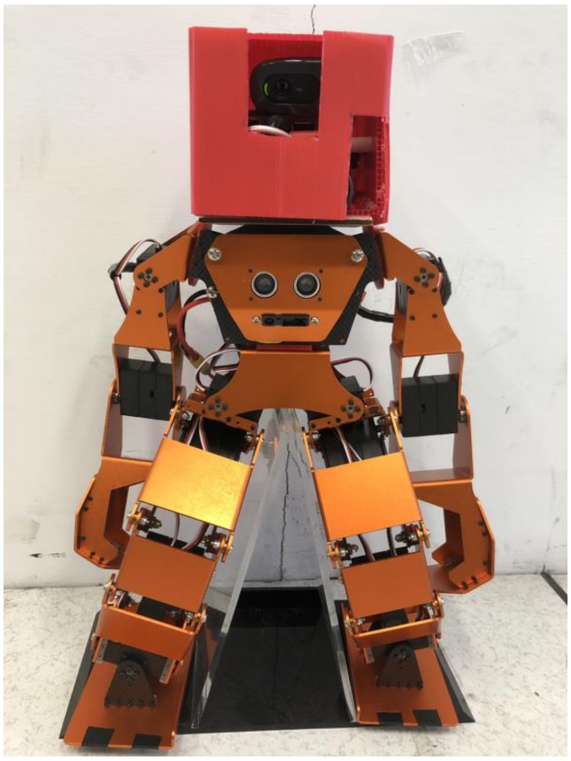


Fig 12. The robot we used in project