

Graduate Project Title: Poor Man Deep Learning Camera

Artificial intelligence (AI) and Block Chain are already affecting our lives in many ways. I envision that they are forming a new virtual world, the robot world. After about 10 years, AI-equipped robotic entities will use bitcoin to trade with the human world and also use bitcoin to trade with other robotic entities. This is one of projects assigned to my first-year graduate students. The Project belonging to an Artificial Intelligence class is helpful to start your AI Experiments with Google AI software Tensorflow. This type of AI Experiments should also be considered as the first step to learn Cloud Machine Learning Engine, i.e., Cloud AutoML. The incoming AutoML helps you easily train high quality custom machine learning models with limited machine learning expertise needed.

The Graduate Project Report is written in Chinese language
by:

Wei JingCheng, Hu Yi, Xu XiaoYu

Email: wjcyx2009@qq.com

875328371@qq.com

2991213620@qq.com

Advisor: Meng QingMin

mengqm@njupt.edu.cn

College of Telecommunication & Information Engineering of
Nanjing University of Posts and Telecommunications, 210003,
Nanjing, P.R. China

2018.1.3-2018.1.12

一. 项目描述^[1-2]

根据亚马逊推出的 DeepLens 原理，通过树莓派以及检测网络模型 YOLO 搭建一个用于检测鸟儿的智能摄像头。它的功能是检测何时有人出现在摄像头的图像中，如果发现图像中存在人，就保存这张照片，并检测图片中人所在位置。为了在摄像机中建立一个深度学习模型，我们使用一台“哑”摄像计算机，将其连接到一个摄像头上，然后通过 WIFI 发送图像。在权衡时延问题后，可以建立一个类似于 DeepLens 产品且更便宜的智能摄像头。将用 Python 编写一个网络服务器，该服务器将树莓派上的图像传送到另外一台计算机上以完成推理或图像检测。

该项目框架如图 1 所示，我们使用树莓派 3b 与树莓派摄像头搭建一台 web 服务器，当在同一局域网中的另一台电脑访问该服务器相应的 URL 时，服务器就会通过摄像头拍取图像，返回一张实时图像。访问者计算机获取图像后对其进行处理分析，判断是否有人的存在。

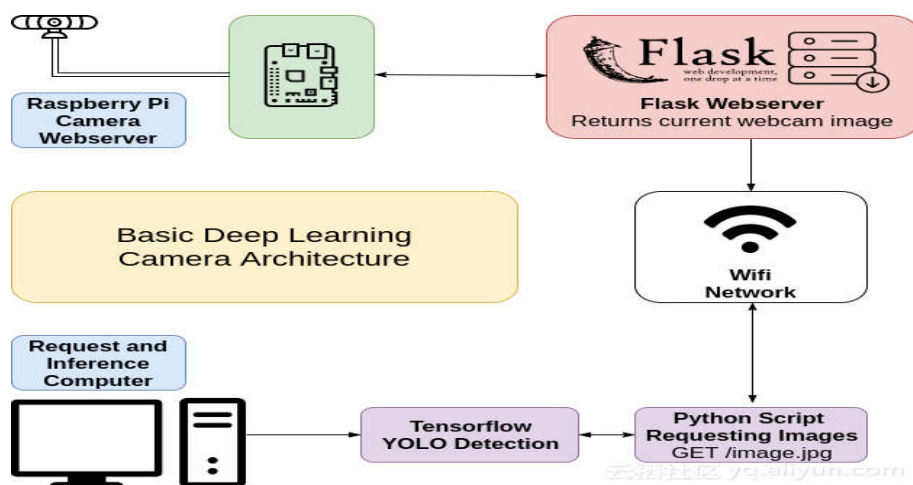


图1 “穷人版”深度学习摄像头项目框架图[1][16]

二. 项目器材

树莓派 3b 一块、树莓派摄像头一个、32G 的 SD（烧写树莓派系统）
一张、网线一根、充电器一个、笔记本电脑两台

三. 任务分配及联系方式

魏景垚：使用树莓派 3b 搭建摄像机 web 服务器

邮箱： wjcyx2009@qq.com

胡翼：搭建虚拟机相关环境配置与扩展库

邮箱： 875328371@qq.com

许晓宇：完成图像识别模块

邮箱： 2991213620@qq.com

四. Web 服务器搭建

1. 系统安装^[3]

所需软件

SD 卡格式化工具：

https://www.sdcard.org/downloads/formatter_4/eula_windows/

树莓派系统：<https://www.raspberrypi.org/downloads/raspbian/>

Win32 Disk Imager：用于将树莓派系统烧写入 SD 卡中

<http://pan.baidu.com/s/1hsi7uRi>

树莓派 3b 需要安装系统才可使用，可以通过向 SD 卡中
烧写系统以完成系统的安装。

首先将 SD 卡插入读卡器中连接电脑 -> 使用 SD 卡格式化工具格式化 SD 卡（如果为新卡就跳过该步） -> 启动 Win32 Disk Imager 软件 -> 将已下载的系统镜像文件导入 -> 然后点击 Write 等待烧写完成。但是，默认的新系统是不开启 SSH 登录服务的，可以在内存卡的 boot 目录下创建一个名为 ssh 的文本文件以实现 SSH 登录。取下 SD 卡，将其插入树莓派 3b 的 SD 卡槽中即可。

2. 连接树莓派并配置^[3-6]

所需软件

PuTTY 软件：用于 SSH 登录树莓派

由于没有树莓派 3b 配套的显示屏，可以通过连接笔记本作为树莓派的显示屏。以 windows10 系统为例，配置方法如下：

1) 将树莓派通电，并用网线将笔记本与树莓派连接，打开电脑的网络和共享中心->更改适配器设置->找到无线网络连接并右键“属性” ->在共享选项卡上选中“允许其他网络用户通过此计算机的 Internet 连接来连接 (N)”选项->确定。如图 2 所示。

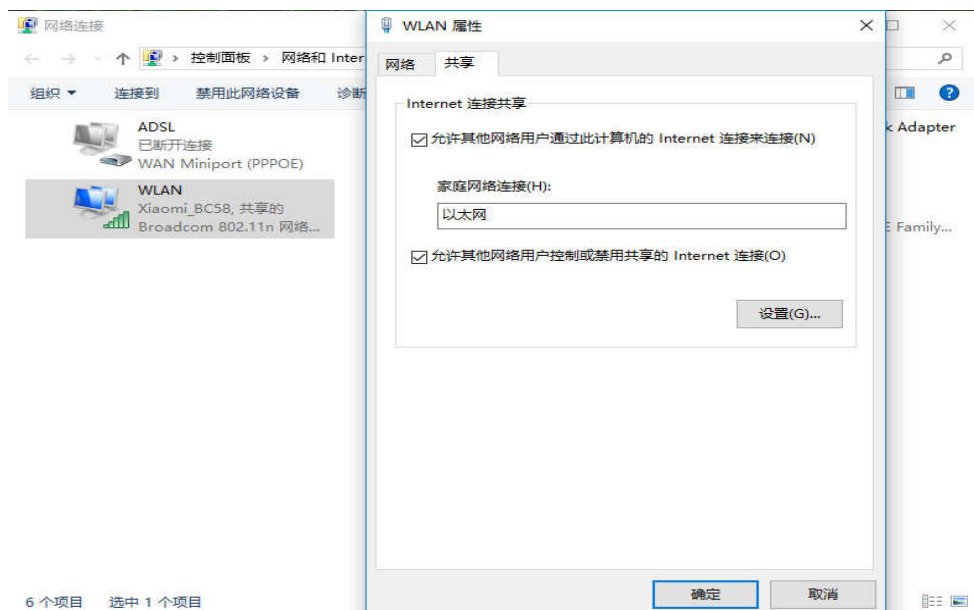


图 2 网络配置

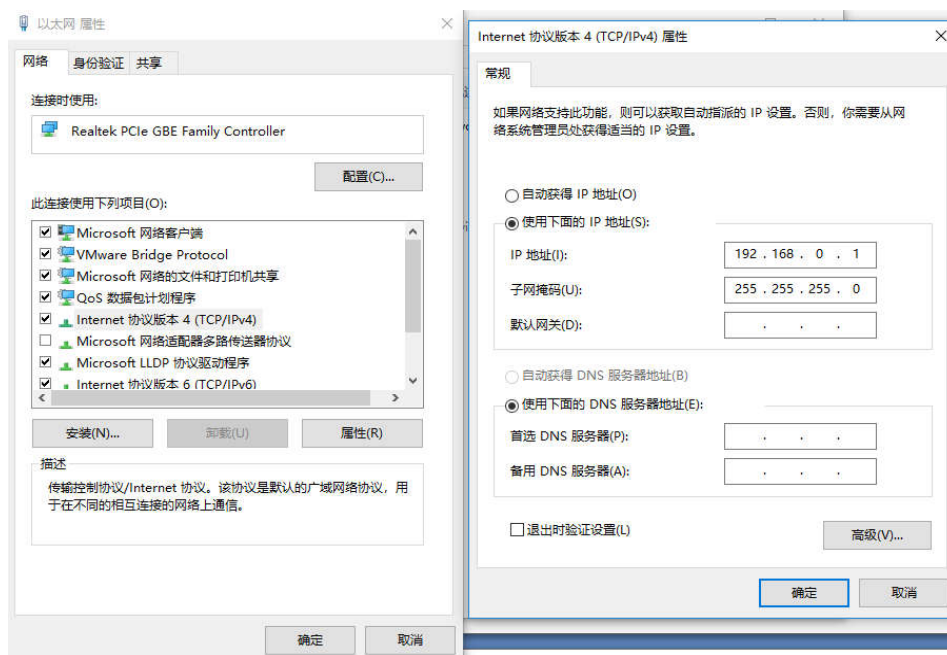


图 3 以太网所分配到的 IP 地址

2) 首先查看以太网所分配到的 IP 地址（如图 3），可知本例的 IP 地址为 192.168.0.1，通过对比插上网线前后的 IP 数量来查找树莓派的 IP 地址(运行 cmd→输入命令 arp -a，如图 4，如果没看到多刷新几次，或者把树莓断电再重新连

接),在接口 192.168.0.1 下会出现树莓派 3b 的 IP 地址(本例树莓派 3b 的 IP 为 192.168.0.235)。

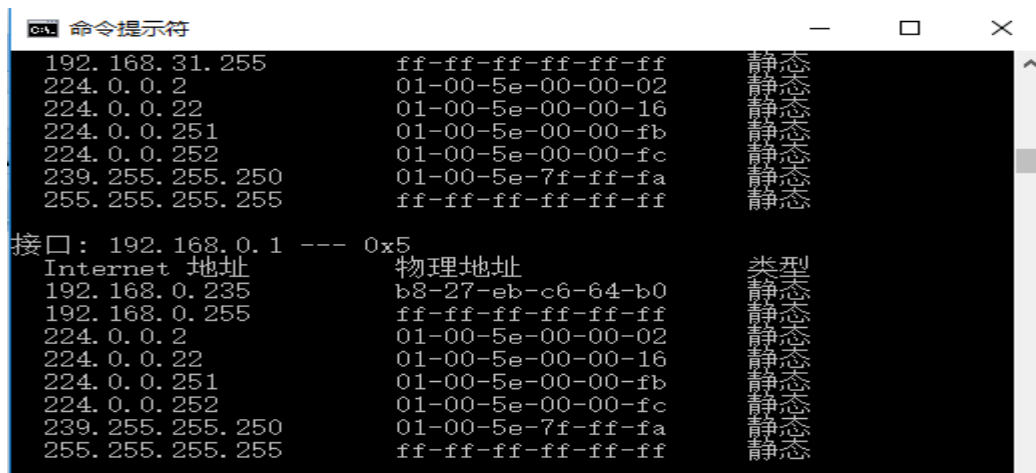


图 4 树莓派 3b 的 IP

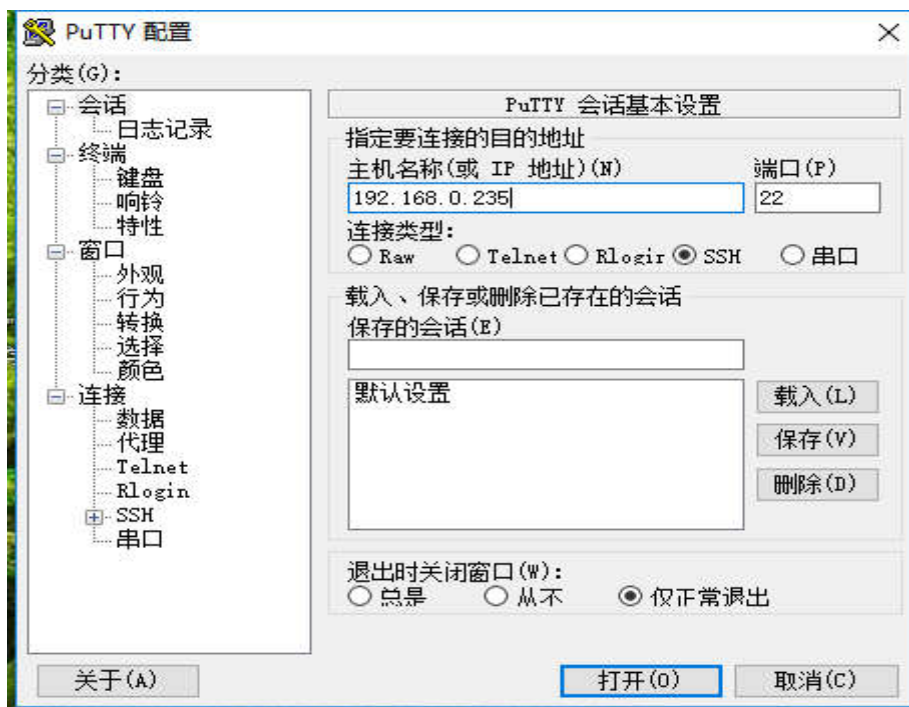
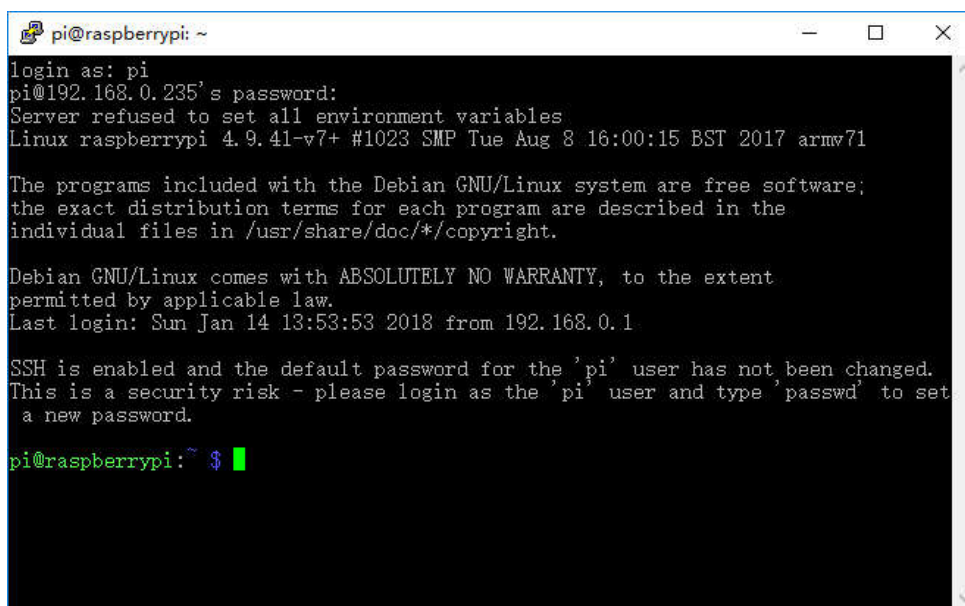


图 5 PuTTY 配置

3) 运行 PuTTY, 将树莓派 3b 的 IP 地址输入, 如图 5 所示。点击“打开”以后会显示登录界面。树莓派的初始账号:pi, 密码:raspberrypi。图 6 所示即为登录成功。



```
pi@raspberrypi: ~
login as: pi
pi@192.168.0.235's password:
Server refused to set all environment variables
Linux raspberrypi 4.9.41-v7+ #1023 SMP Tue Aug 8 16:00:15 BST 2017 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jan 14 13:53:53 2018 from 192.168.0.1

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.
pi@raspberrypi:~ $
```

图 6 登录树莓派

4) 由于可视界面操作树莓派更加方便，可以会用 Windows 系统自带的远程桌面连接程序连接树莓派，方法为：在图 6 中输入命令 `sudo apt-get install tightvncserver`，安装 `tightvncserver`，安装完毕后输入命令 `sudo apt-get install xrdp`，进行 `xrdp` 服务的安装。安装完成后，输入一下命令：

```
sudo /etc/init.d/xrdp start
```

```
sudo update-rc.d xrdp defaults
```

第一条命令是启动 `xrdp` 服务，第二条命令是将 `xrdp` 服务加入到系统默认启动服务中。

5) 启动 windows 电脑中的远程桌面控制程序，输入树莓派的 IP，启动后，输入树莓派的账号与密码即可连接至树莓派桌面。如图 7、图 8 和图 9 所示。

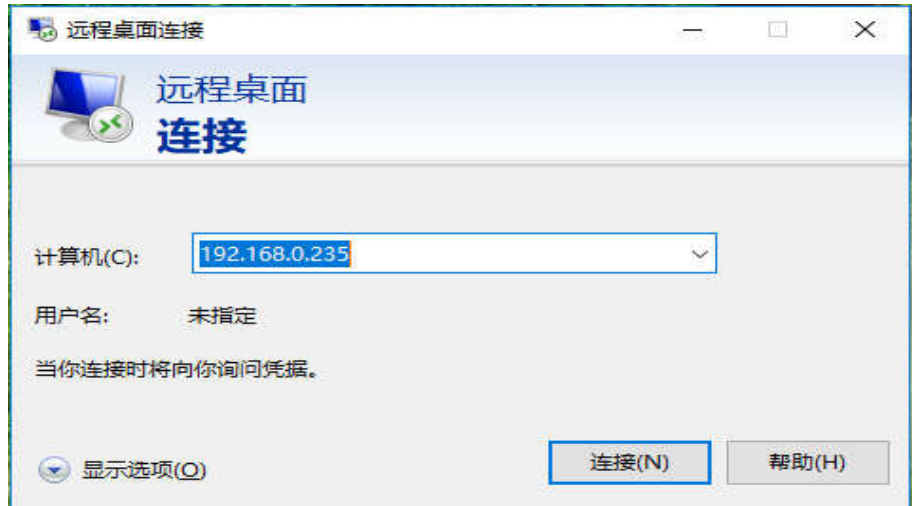


图 7 远程连接树莓派

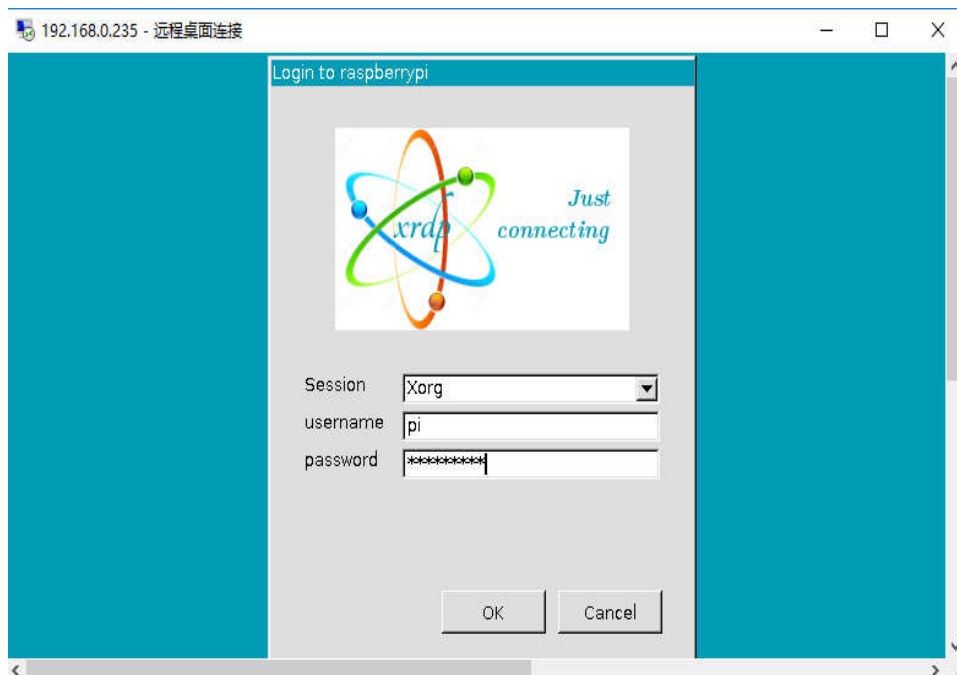


图 8 登录树莓派

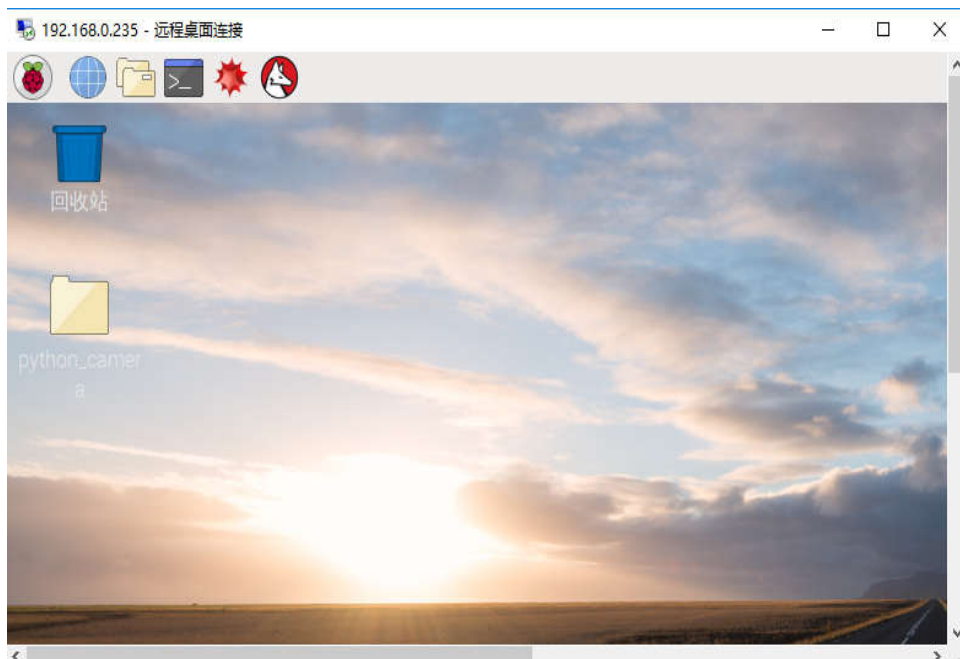


图 9 树莓派界面

3. 搭建 web 服务器^[2]

由于树莓派系统中自带 Python 与 IDLE，故我们可以直接使用创建 Python 的 web 服务器工程(工程的全部程序见附录)，本文初步使用的摄像头服务器代码为 Miguel Grinberg 的经典服务器代码，如图 10 所示

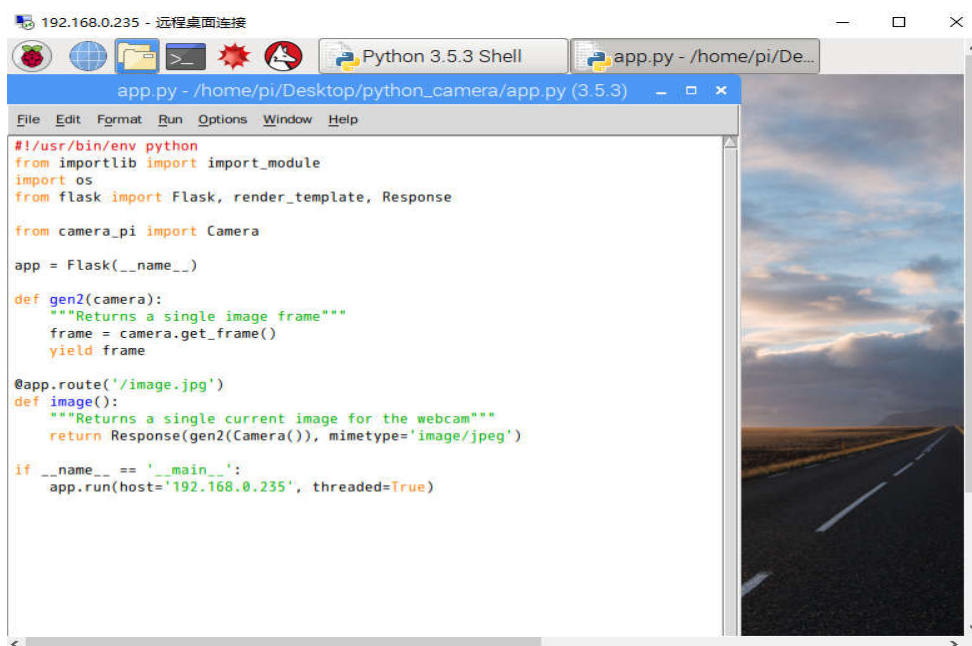


图 10 Miguel Grinberg 的经典服务器代码

需要注意的是将自己树莓派的 IP 地址替换上去。运行后就可
在相应的 URL 中访问得到摄像头所拍摄的图片，本例中的
URL 为：<https://192.168.0.235:5000/image.jpg>，如图 11
所示。另外，我们还可以通过在 windows 界面下开启 WiFi，
凡是连接至此 WiFi 的电脑皆可访问树莓派 web 服务器的 URL
获取图片。

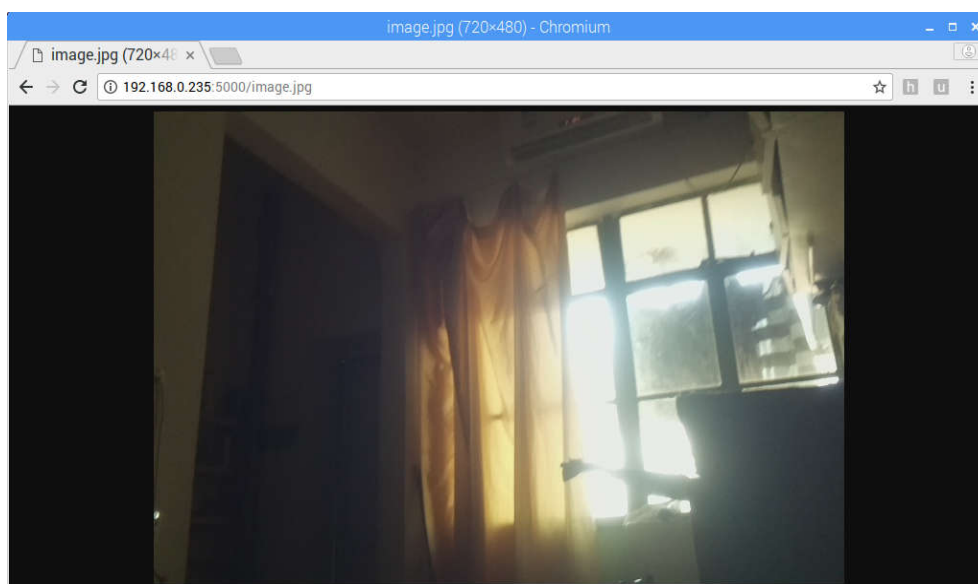


图 11 访问树莓派服务器

五. 虚拟机的安装与相关设置

1. 系统安装^[7-9]

所需软件：

VMware workstation Pro [for windows]

<https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>

ubuntu 16.04 LTS 镜像文件

<https://www.ubuntu.com/download/desktop/contribute?version=16.04.3&architecture=amd64>

vmware workstation Pro

安装版本: 14.1.1

密钥: CV7T2-6WY5Q-48EWP-ZXY7X-QGUWD

将 vmware 安装完成后 -> 打开 -> 创建新的虚拟机->下一步

->稍后安装操作系统(s)->Linux(L)->下一步

名称设为:HY

磁盘分配空间设置为: 40G

完成后, 如图 12 所示, 设置虚拟机内存

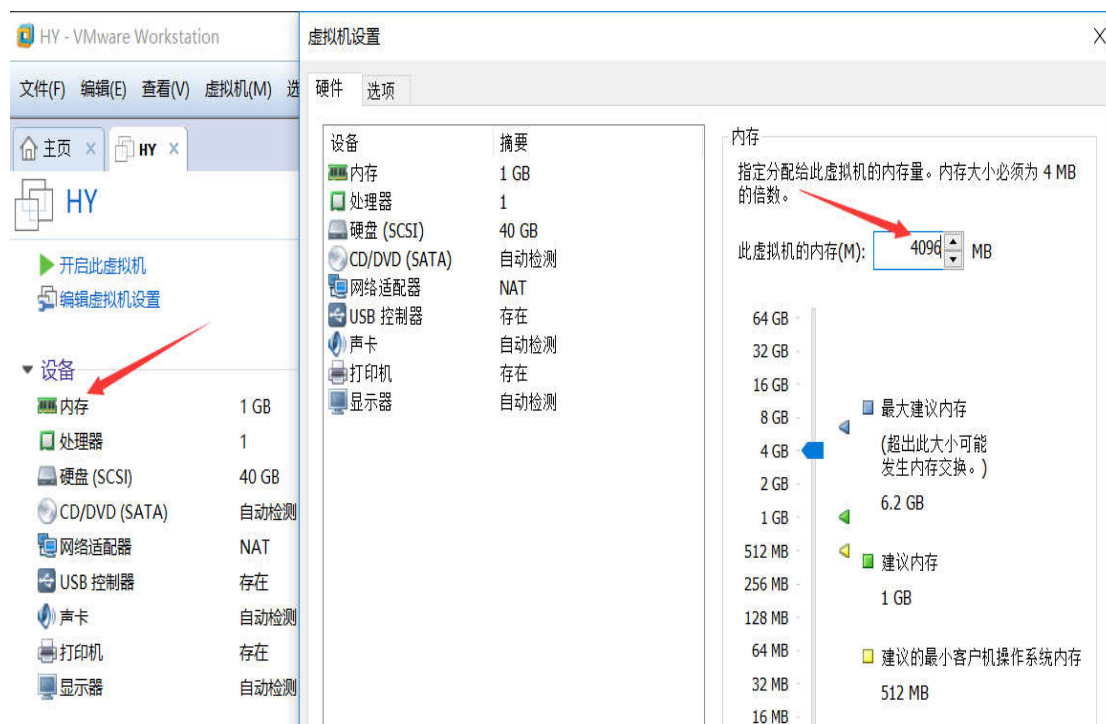


图 12 虚拟机内存设置

其次导入镜像文件 Ubuntu 16.04 LTS, 如图 13 所示



图 13 镜像文件的导入

开启虚拟机【确保电脑 BIOS 已经设置开启虚拟】->选择中文简体语言->安装 Ubuntu->继续->清除整个磁盘并安装 Ubuntu->现在安装->继续->shanghai->继续->设置姓名 (HY)、密码、自动登录。



图 14 系统安装图

2. 相关设置

1) 屏幕设置^[9]

由于操作桌面不能全屏，可视化不方便。即可通过以下操作设置，打开 vm 的虚拟机选项卡->安装 VMware Tools->是->点开系统左边图标 VMware Tools->将其文件夹下的五个文件全部拷贝到本地【下载目录下】->把 VMwareTools 的 tar.gz 文件解压到下载文件夹内->进入解压后的文件夹内->打开终端运行：`sudo perl vmware-install.pl` -> y 一直运行下去即可，完成后关机重启

2) 浏览器默认设置

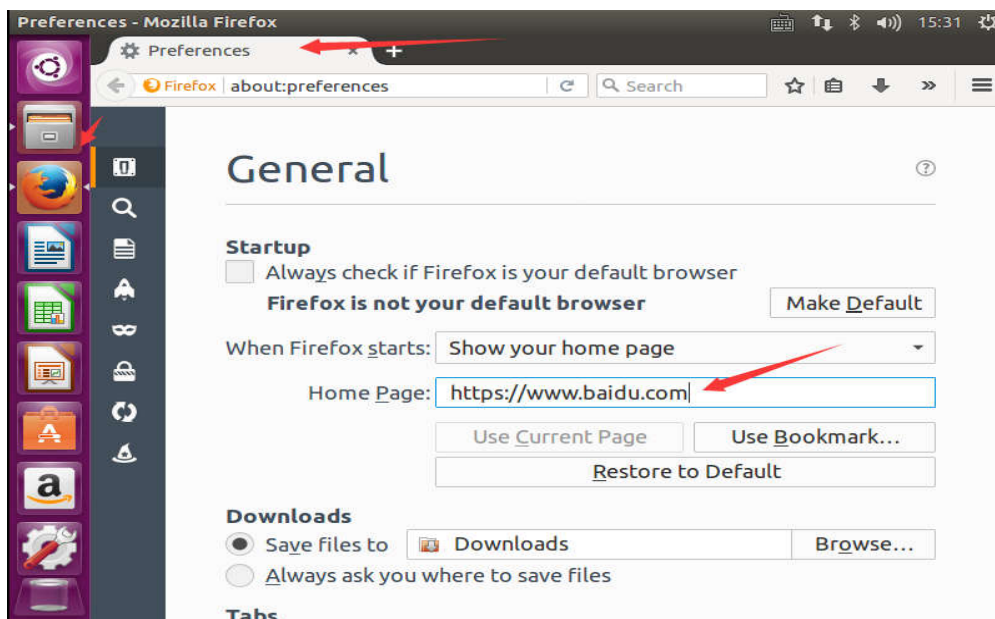


图 15 浏览器默认设置

3) 系统设置更新

进入系统设置->系统->软件和更新->下载来自中国的服务器->其它站点->选择 aliyun->重新载入->等待更新缓存.
如图 16 所示:

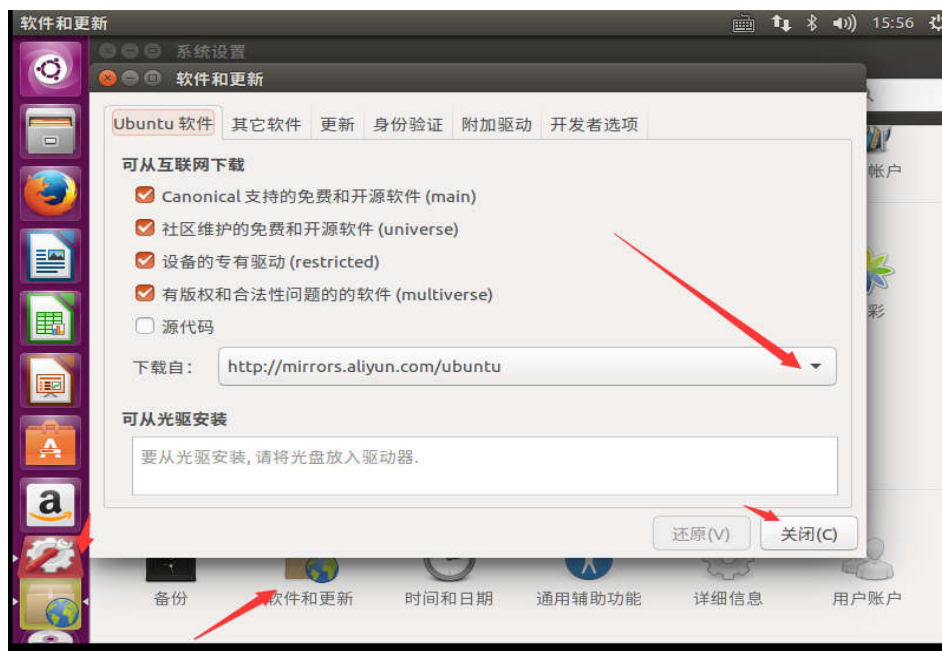


图 16 系统设置

六. 项目运行相关库及环境配置

所需安装包

cmake-3.10.1.tar.gz: <https://cmake.org/download/>

opencv3.2.0:

<https://codeload.github.com/opencv/opencv/zip/3.2.0>

所需文件

ippicv_linux_20151201: <https://pan.baidu.com/s/1htcCNUc>

密码: 71pr

weights 文件: <https://pjreddie.com/darknet/yolo/>

1. cmake 的安装^[10]

网页直接打开下载地址下载->解压下载后的文件->进入解压

后的 cmake-3.10.1 文件夹->打开终端依次执行以下命令

1). /bootstrap

2) make -j8

3) sudo make install

2. opencv3.2.0 的下载安装^[11]

1) 进入 root 权限^[12]: sudo -i

2) 对 Ubuntu apt-get 仓库进行更新

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

运行第二句时会出现有些软件包无法下载，此时退出 root 权限，重新打开终端不使用 root 权限再运行 sudo apt-get upgrade

3) 安装必要的 python 插件与环境

```
sudo -i
```

```
sudo apt-get install python3-setuptools python3-dev -y
```

4) 安装 pip

```
sudo easy_install3 pip
```

5) 更新 pip(非必要步骤)

```
pip install --upgrade pip
```

6) 安装 numpy 包

```
pip install numpy
```

7) 安装 build-essential

```
sudo apt-get install build-essential -y
```

8) 安装其他开发包

```
sudo apt-get install cmake
```

```
sudo apt-get install git
```

```
sudo apt-get install libgtk2.0-dev
```

```
sudo apt-get install pkg-config
```

```
sudo apt-get install libavcodec-dev
```



```
sudo apt-get install libavformat-dev
```

```
sudo apt-get install libswscale-dev -y
```

若以上某句安装过程中, 出现有几个软件无法安装, 再次运行命令。

10) 下载 opencv3.2.0^[12]

选择本地一个文件夹(下载)->进入下载文件夹内新建一个名为 opencv 的新的文件夹->打开终端运行

```
wget https://codeload.github.com/opencv/opencv/zip/3.2.0
```

```
unzip opencv-3.2.0.zip && cd opencv-3.2.0
```

```
mkdir build && cd build
```

再运行以下命令[确保命令等号右边的路径均存在]:

```
cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr
/local PYTHON3_EXECUTABLE=/usr/bin/python3 PYTHON_INCLUDE_DIR=
/usr/include/python3.5 PYTHON_LIBRARY=/usr/lib/x86_64-linux-gn
u/libpython3.5m.so PYTHON3_NUMPY_INCLUDE_DIRS=/usr/local/lib/p
ython3.5/dist-packages/numpy/core/include ..
```

运行完上述命令后, 会出现下载 ippecv 的失败, 此时在路径:

下载/opencv/opencv-3.2.0/3rdparty/ippicv/downloads/li
nux-808b791a6eac9ed78d32a7666804320e 下个不完整的 oppi
cv 文件, 将此文件删除->下载 ippicv_linux_20151201 放在路
径下载/opencv/opencv-3.2.0/3rdparty/ippicv/downloads/
linux-808b791a6eac9ed78d32a7666804320e/的文件夹下->再
在 build 文件夹下打开终端运行上述命令 -> 运行 make && s
udo make install , 等待安装完毕。

3. 下载安装 darkflow^[13-14]

重新打开终端，进入 root 权限

```
sudo -i
```

```
sudo pip3 install Cython --install-option="--no-cython-compile"
```

```
git clone https://github.com/thtrieu/darkflow
```

```
cd darkflow
```

```
python3 setup.py build_ext --inplace
```

```
pip3 install .
```

出现错误

解决： `sudo python3 setup.py install`

```
pip3 install .
```

4. 安装 tensorflow

```
sudo -i
```

```
pip3 install tensorflow
```

5. 配置程序中 weights 文件路径^[15]

下载 weights 文件->放在本地下载文件夹 (/home/hy/下载/)
->把 predict.py 和 predict_draw.py 程序中 cfg 文件路径改
为其完整路径 (home/hy/下载/darkflow/cfg/), weights 文件
读取路径也改为其存放的路径。

七. 项目实施与解读

程序 predict.py 如图 17 所示

```
from darkflow.net.build import TFNet
import cv2

from io import BytesIO
import time
import requests
from PIL import Image
import numpy as np

options = {"model": "/home/hy/TF/darkflow/cfg/tiny-yolo-voc.cfg", "load": "/home/hy/TF/tiny-yolo-voc.weights", "threshold": 0.1}
tfnet = TFNet(options)

personSeen = 0
def handleBird():
    pass

while True:
    r = requests.get('http://192.168.1.1:5000/image.jpg')_# replace with your ip address
    curr_img = Image.open(BytesIO(r.content))
    curr_img_cv2 = cv2.cvtColor(np.array(curr_img), cv2.COLOR_RGB2BGR)

    # uncomment below to try your own image
    #imgcv = cv2.imread('./sample/bird.png')
    result = tfnet.return_predict(curr_img_cv2)
    #print(result)
    for detection in result:
        if detection['label'] == 'person':
            print('person detected')
            personSeen += 1
            curr_img.save('person/%i.jpg' % personSeen)
    print('running again')
    time.sleep(4)
```

图 17 predict.py 程序

程序解读：option 读取之前下载对应路径的 cfg、weights 文件，运行 predict.py 程序，则其会去访问我们通过树莓派搭建的服务器地址，从该处获取此刻树莓派摄像头图片内容，如果检测到图片中含有人，则程序启动保存命令，将图片保存到 person 文件夹内，如果此刻未检测到有人，则不保存图片，程序显示 run again，四秒钟后，再次访问服务器，依此循环检测，只有终止程序才会停止访问服务器。实验访问一次服务器所得结果如图 18 所示



图 18 predict.py 运行检测图

程序 predict_draw.py 程序如图 19 所示

```
1 from darkflow.net.build import TFNet
2 import cv2
3
4 from io import BytesIO
5 import time
6 import requests
7 from PIL import Image, ImageDraw
8 import numpy as np
9
10
11 import glob
12
13 options = {"model": "/home/hy/TF/darkflow/cfg/tiny-yolo-voc.cfg", "load": "/home/hy/TF/darkflow/weights/tiny-yolo-voc.weights", "threshold": 0.2}
14
15 tfnet = TFNet(options)
16
17 counter = 0
18 for filename in glob.glob('person/*.jpg'):
19     curr_img = Image.open(filename).convert('RGB')
20     curr_imgcv2 = cv2.cvtColor(np.array(curr_img), cv2.COLOR_RGB2BGR)
21
22     result = tfnet.return_predict(curr_imgcv2)
23     print(result)
24     draw = ImageDraw.Draw(curr_img)
25     for det in result:
26         draw.rectangle([det['topleft']['x'], det['topleft']['y'],
27                        det['bottomright']['x'], det['bottomright']['y']],
28                        outline=(255, 0, 0))
29         draw.text([det['topleft']['x'], det['topleft']['y'] - 25], det['label'], fill=(0, 10, 100))
30     curr_img.save('person_labeled/%i.jpg' % counter)
31     counter += 1
```

图 19 predict_draw.py 程序图

程序解读：predict_draw.py 主要是对保存的图片进行识别并对识别物进行框图显示保存。图片识别程序通过我们之前导入的已经训练好的 YOLO 模型的 weights 文件，调整识别的阈值，对已经保存在 person 文件夹内的图片依次进行识别，识别后，程序会对其通过红色框图进行标记，并且会在框图的左上方标记识别物名称，其次把这样处理后的图片保存再 person_labeled 文件夹内依次保存。效果如图 20 所示



图 20 predict_draw.py 程序运行图片识别效果图

八. 项目程序、硬件图片附录

1. 树莓派 3b 构建 web 服务器工程

工程包含三个文件：[camera_pi.py](#), [base_camera.py](#), [app.py](#)

base_camera.py

```
import time
import threading

try:
    from greenlet import getcurrent as get_ident
except ImportError:
    try:
        from thread import get_ident
    except ImportError:
        from _thread import get_ident

class CameraEvent(object):
    """An Event-like class that signals all active clients when a new frame is
    available.
    """
    def __init__(self):
        self.events = {}
    def wait(self):
        """Invoked from each client's thread to wait for the next frame."""
        ident = get_ident()
        if ident not in self.events:
            # this is a new client
            # add an entry for it in the self.events dict
            # each entry has two elements, a threading.Event() and a timestamp
            self.events[ident] = [threading.Event(), time.time()]
        return self.events[ident][0].wait()
    def set(self):
        """Invoked by the camera thread when a new frame is available."""
        now = time.time()
        remove = None
        for ident, event in self.events.items():
            if not event[0].isSet():
                # if this client's event is not set, then set it
                # also update the last set timestamp to now
                event[0].set()
                event[1] = now
            else:
                # if the client's event is already set, it means the client
```

```

        # did not process a previous frame
        # if the event stays set for more than 5 seconds, then assume
        # the client is gone and remove it
        if now - event[1] > 5:
            remove = ident

    if remove:
        del self.events[remove]

    def clear(self):
        """Invoked from each client's thread after a frame was processed."""
        self.events[get_ident()][0].clear()

class BaseCamera(object):
    thread = None # background thread that reads frames from camera
    frame = None # current frame is stored here by background thread
    last_access = 0 # time of last client access to the camera
    event = CameraEvent()

    def __init__(self):
        """Start the background camera thread if it isn't running yet."""
        if BaseCamera.thread is None:
            BaseCamera.last_access = time.time()

            # start background frame thread
            BaseCamera.thread = threading.Thread(target=self._thread)
            BaseCamera.thread.start()

            # wait until frames are available
            while self.get_frame() is None:
                time.sleep(0)

    def get_frame(self):
        """Return the current camera frame."""
        BaseCamera.last_access = time.time()

        # wait for a signal from the camera thread
        BaseCamera.event.wait()
        BaseCamera.event.clear()

        return BaseCamera.frame

    @staticmethod
    def frames():
        """Generator that returns frames from the camera."""
        raise RuntimeError('Must be implemented by subclasses.')

    @classmethod
    def _thread(cls):
        """Camera background thread."""
        print('Starting camera thread.')

        frames_iterator = cls.frames()
        for frame in frames_iterator:
            BaseCamera.frame = frame
            BaseCamera.event.set() # send signal to clients

```

```
time.sleep(0)

# if there hasn't been any clients asking for frames in
# the last 10 seconds then stop the thread
if time.time() - BaseCamera.last_access > 10:
    frames_iterator.close()
    print('Stopping camera thread due to inactivity.')
    break

BaseCamera.thread = None
```

camera_pi.py

```
import io
import time
import picamera
from base_camera import BaseCamera

class Camera(BaseCamera):
    @staticmethod
    def frames():
        with picamera.PiCamera() as camera:
            # let camera warm up
            time.sleep(2)
            stream = io.BytesIO()
            for foo in camera.capture_continuous(stream, 'jpeg', use_video_port=True):
                # return current frame
                stream.seek(0)
                yield stream.read()
                # reset stream for next frame
                stream.seek(0)
                stream.truncate()
```

app.py

```
#!/usr/bin/env python

from importlib import import_module
import os

from flask import Flask, render_template, Response
from camera_opencv import Camera

app = Flask(__name__)

def gen2(camera):
    """Returns a single image frame"""
    frame = camera.get_frame()
    yield frame
```

```

@app.route('/image.jpg')
def image():
    """Returns a single current image for the webcam"""
    return Response(gen2(Camera()), mimetype='image/jpeg')
if __name__ == '__main__':
    app.run(host='0.0.0.0', threaded=True)

```

2. 图片识别工程

工程包含 2 个文件: [predict.py](#), [predict_draw.py](#)

predict.py

```

from darkflow.net.build import TFNet
import cv2
from io import BytesIO
import time
import requests
from PIL import Image
import numpy as np

options = {"model": "/home/hy/下载/darkflow/cfg/tiny-yolo-voc.cfg", "load": "/home/hy/下载/tiny-yolo-voc.weights", "threshold": 0.1}
tfnet = TFNet(options)
personSeen = 0
def handleBird():
    pass
while True:
    r = requests.get('http://192.168.1.1:5000/image.jpg') # replace with your ip address
    curr_img = Image.open(BytesIO(r.content))
    curr_img_cv2 = cv2.cvtColor(np.array(curr_img), cv2.COLOR_RGB2BGR)
    # uncomment below to try your own image
    #imgcv = cv2.imread('./sample/bird.png')
    result = tfnet.return_predict(curr_img_cv2)
    #print(result)
    for detection in result:
        if detection['label'] == 'person':
            print("person detected")
            personSeen += 1
            curr_img.save('person/%i.jpg' % personSeen)
    print('running again')
    time.sleep(4)

```


predict_draw.py

```
from darkflow.net.build import TFNet
import cv2
from io import BytesIO
import time
import requests
from PIL import Image, ImageDraw
import numpy as np
import glob

options = {"model": "/home/hy/下载/darkflow/cfg/tiny-yolo-voc.cfg", "load": "home/hy/下载/
/tiny-yolo-voc.weights", "threshold": 0.15}
tfnet = TFNet(options)
counter = 0
for filename in glob.glob('person/*.jpg'):
    curr_img = Image.open(filename).convert('RGB')
    curr_imgcv2 = cv2.cvtColor(np.array(curr_img), cv2.COLOR_RGB2BGR)
    result = tfnet.return_predict(curr_imgcv2)
    print(result)
    draw = ImageDraw.Draw(curr_img)
    for det in result:
        draw.rectangle([det['topleft']['x'], det['topleft']['y'],
                        det['bottomright']['x'], det['bottomright']['y']],
                        outline=(255, 0, 0))
        draw.text([det['topleft']['x'], det['topleft']['y'] - 13], det['label'], fill=(0, 10,
100))
    curr_img.save('person_labeled/%i.jpg' % counter)
    counter += 1
```

3. 树莓派硬件实物图



图 21 低成本的树莓派硬件实物图

九. 参考文献

- [1] Jason Brownlee 著. 海棠 译. 使用树莓派和 YOLO——打造一个“穷人版”深度学习摄像头[DB/OL]. <https://www.jianshu.com/p/1d3648cb285d>, 2017.
- [2] burningion. poor-mans-deep-learning-camera[DB/OL]. <https://github.com/burningion/poor-mans-deep-learning-camera>, 2017.
- [3] 科技爱好者. 无显示器通过网线连接笔记本电脑玩转树莓派[DB/OL]. <https://www.jianshu.com/p/505fc3f957aa>, 2017.
- [4] 战侠歌 1994. (SSH+VNC) 树莓派一根网线连电脑, 不要显示屏[DB/OL]. <https://www.cnblogs.com/zhanxiage1994/p/4096537.html>, 2014.
- [5] 实习生 76. 没有显示器, 树莓派安装系统并连接到笔记本电脑[DB/OL]. http://blog.csdn.net/qq_27774983/article/details/52858592, 2016.
- [6] admin. 用 Windows 远程桌面连接树莓派的方法[DB/OL]. <http://shumeipai.nxez.com/2013/10/06/windows-remote-desktop-connection-raspberry-pi.html>, 2013.
- [7] ClanS. VMware 虚拟机安装 Ubuntu 简单教程[DB/OL]. <https://www.jianshu.com/p/3379892948da>, 2016.
- [8] zd423. VMware Pro v14.1.1 官方版本及激活密钥[DB/OL]. <http://www.zdfans.com/5928.html>, 2018.
- [9] 牧野. VMware 虚拟机 ubuntu 显示屏幕太小解决办法[DB/OL]. <http://blog.csdn.net/dcrmg/article/details/74090307>, 2017.
- [10] 鲁提辖_Ga. ubuntu 安装 CMake 的几种方式[DB/OL]. <http://blog.csdn.net/lj402159806/article/details/76408597>, 2017.
- [11] 你微笑很美. Ubuntu16.04/树莓派 Python3+opencv 配置[DB/OL]. http://blog.csdn.net/qq_37910312/article/details/72866242, 2017.
- [12] neilooo. ubuntu 编译安装 opencv3.3+python3, 解决 ipicv 下载问题[DB/OL]. <http://blog.csdn.net/neilooo/article/details/78425559>, 2017.
- [13] imperfect00. darkflow 测试和训练 yolo[DB/OL]. <http://blog.csdn.net/u011961856/article/details/76582669>, 2017.
- [14] 阳小良. ubuntu 权限不够(进入后身份并不是 root 而是自己的默认登录名的情况)[DB/OL]. <http://blog.csdn.net/devilzy2656/article/details/8235428>, 2012.
- [15] Joseph chet Redmon. YOLO:Real-Time Object Detection[DB/OL]. <https://pjreddie.com/darknet/yolo/>, 2017.

[16]

https://r.search.yahoo.com/_ylt=A0LEV2PRkWWaMcAALVwj4gt.;_ylu=X3oDMTB yOHZyb2ltBGNvbG8DYmYxBHBvcwMxBHZ0aWQDBHNlYwNzcg--/RV=2/RE=1516634705/R0=10/RU=https%3a%2f%2finfozonic.com%2f2017%2f12%2f19%2fbuilding-a-poor-mans-deep-learning-camera-in-python%2f/RK=2/RS=rVehHdECULyXFXRugKUnxzbiz4Q-

请引用该报告。

Please quote the report:

Meng Qingmin, Wei JingCheng, Hu Yi, Xu Xiaoyu. Poor Man Deep Learning Camera. NUPT, 2018. 1. <https://github.com/462124653/AIExp.git>