

GNG5140 Report Deliverable F

[Revised User Manual]

Submitted by

[Group 3 – Environmental Robot]

[Qun Cao, 300091704]

[Yuxuan Li, 300070573]

[Shiyao Xu, 300070250]

April 5, 2020

University of Ottawa

Abstract

Environmental problems have increasingly become the focus of attention. Because of the concern of pollution on the seashore, our project aims to design and modify a robot for beach cleaning. According to the requirement of clients, benchmarking and the experience during trails, we designed a mobile platform for camera to make it flexible and get larger vision. Meanwhile, we also developed the night work mode based on IR camera to make it able to work automatically at night. Our prototype and functional testing are finished as well as the analysis of usability, scalability, sustainability and other necessary analysis. User manual, conclusion and recommendation for future work are introduced detailed in this report.

Table of Contents

- Abstract..... i
- Table of Contents ii
- List of Figures ii
- List of Acronyms iv
- 1 Introduction..... 5
- 2 Main Subject Body of the Report 6
 - 2.1 User Manual..... 6
 - 2.1.1 Important Features, Functions and Capabilities..... 6
 - 2.1.2 Detailed Instructions for Camera Arm..... 7
 - 2.1.3 Detailed Instructions for Object Detection 10
 - 2.1.4 Safety Guidelines 14
 - 2.1.5 Troubleshooting and Technical Instruction 14
 - 2.2 Design File 15
- 3 Conclusions and Recommendations for Future Work 16
- 4 Bibliography 17
- APPENDICES 18

List of Figures

Figure 1 3Dmodel of Bowie and the camera arm	7
Figure 2 The whole frame, camera arm and base	7
Figure 3 Electric components (1~3).....	8
Figure 4 Installation process of the camera arm	8
Figure 5 Installation process of the base.....	9
Figure 6 Controller of Bowie	9
Figure 7 How the camera moves	10
Figure 8 Enable the camera.....	11
Figure 9 tflite1 directory	12
Figure 10 Download the model.....	13
Figure 11 TensorFlow configuration	13
Figure 12 Video input element and connection to TensorFlow.....	13
Figure 13 Object detection result example	14

List of Acronyms

Acronym	Definition
DOF	Degree of freedom
IR camera	Infrared camera
tflite	TensorFlow Lite

1 Introduction

Environmental problems have increasingly become the focus of attention. Because of the concern of pollution on the seashore, our project aims to improve natural space in a more efficient and intelligent way with robots.

Low-cost robots are designed for environmental observation, cleaning, and repair. The first application of environmental robots is reducing the accumulation of small-sized pollution on shorelines. It not only can reduce the cost of human resources, but also can access more locations due to it being less intrusive when compared to large machinery. We improve the robot's design to make sure the camera is able to see different areas, which will greatly improve the working efficiency of the robot. And we also develop the night work mode, which will greatly reduce the impact of the robot on the living environment of other creatures.

Our client is Robot Missions, an organization dedicated to improving the environment. It provides Bowie, the environmental robot platform to everyone who is interested in robots and environment. Before we started the project, Bowie was already able to finish the beach cleaning missions under control. But its camera was fixed and it was a normal RGB camera. The area it could see was limited and it couldn't work at night. In order to improve Bowie, we were required to solve these 2 problems and now we did it.

Our design includes hardware and software parts. We choose cheap but nice components and make the structure easy to assemble or disassemble. The program is effective. After training, the robot can detect objects without light accurately.

2 Main Subject Body of the Report

2.1 User Manual

2.1.1 Important Features, Functions and Capabilities

The original Bowie model can work in supervised and unsupervised situations. Based on the existing model, in order to improve the working efficiency, reduce the control difficulty, and help the robot adapt to different working environments, we provide the following two novel features.

The first one is that the final prototype has a new camera arm with 2 degrees of freedom. It is installed on the right front of the existing robot body, and allows the camera to see different areas and recognize garbage. The new camera arm contains 2 servos. One is used to swing the arm from 90 to 180 degrees and the other is to rotate the camera from 0 to 180 degrees.

The second one is that the final prototype uses the IR camera instead of the normal RGB camera used in the previous model to achieve the function of working without light at night. The final prototype also introduces a different method to store and process captured images. In the final prototype, TensorFlow is installed on Raspberry Pi, connecting Raspberry Pi and IR camera. After configuration, the picture captured by IR camera can be stored in Raspberry Pi. Then in the processing stage, a new pre-trained TensorFlow-based object detection model can be applied to the captured IR images to recognize garbage in real-time. The pre-trained model we use is trained on COCO dataset, this model can recognize more than 70 different things, including bottles, spoons, pizza, etc. The overall accuracy is about 80%, but it varies from detection targets.

For the use of the final prototype, the user can control the movement of the robot through the controller, and then use the second mode of the controller to control the camera's angle. After the robot acquires the corresponding image, it analyzes through the object detection model and then automatically scoop up the identified garbage. Through this new working mode, the more complicated operation of manipulating the robot arm to shovel the garbage through the controller is avoided, which is replaced by the easier operation of manipulating the camera angle of view, and then the robot automatically shovels the garbage. This method combines the principles of supervised work and unsupervised work, which reduces the accuracy required for operation, thereby reducing the difficulty of user operations. Besides, if the robot works in a completely unsupervised situation, the newly added camera arm can also improve the efficiency of the robot. For example, if there is garbage on the left or right area of the robot's movement direction, the robot cannot recognize the garbage by using the rotation method and then scoop it up when it first approaches this place. And the IR camera used in the final prototype enables the robot to work normally at night without adding additional light sources. This ensures that the normal night activities of other creatures on the beach, like turtles, are not affected.

2.1.2 Detailed Instructions for Camera Arm

The camera arm includes a 3D printed frame and some electric components, such as servos and IR camera.

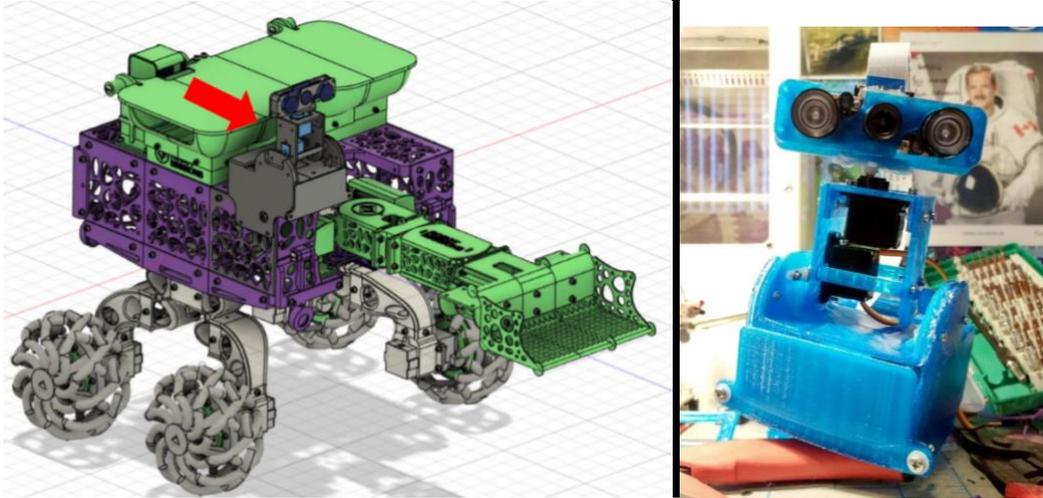


Figure 1 3Dmodel of Bowie and the camera arm

1) Components

i. 3D printed components

The frame is divided into 2 parts: the camera arm and the base. The camera arm is equipped with the servos and the IR camera; the base is used to connect the camera arm and the robot body and hide the Raspberry Pi.

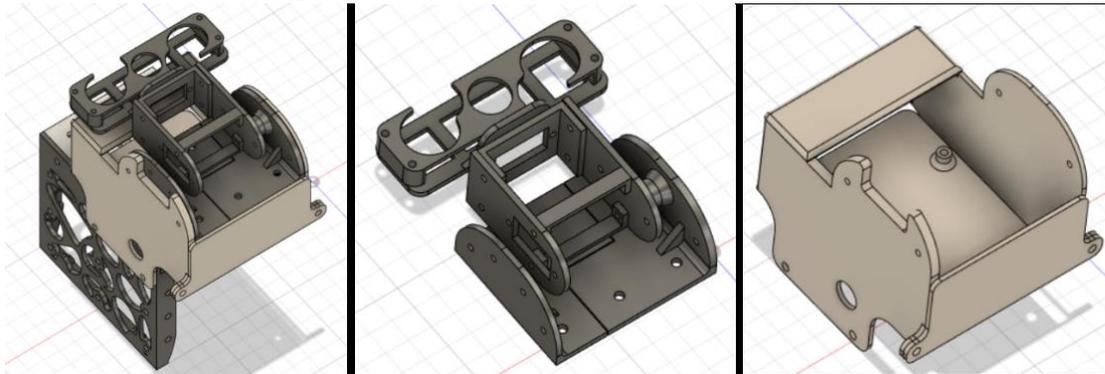


Figure 2 The whole frame, camera arm and base

ii. Electric components

- SG90 Micro Servo Motor (*2)
- Infrared Night Vision Camera for Raspberry Pi (*1)
- Raspberry Pi (*1)



Figure 3 Electric components (1~3)

2) Installation

M2.5 Screws are needed to fix every part.

i. Camera arm

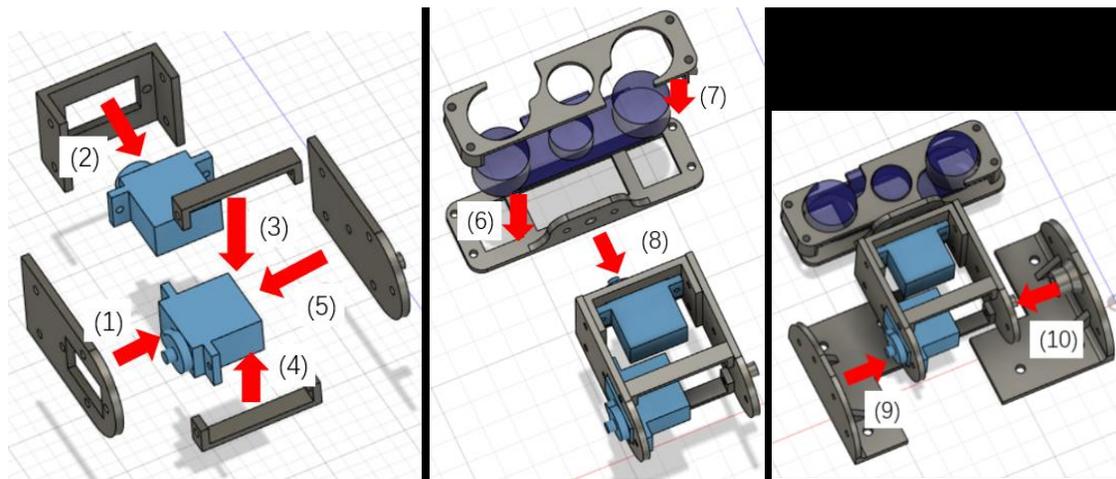


Figure 4 Installation process of the camera arm

ii. Base

- Attach the base shell to the robot body.
- Attach the Raspberry Pi to the board, and then push them into the shell.
-

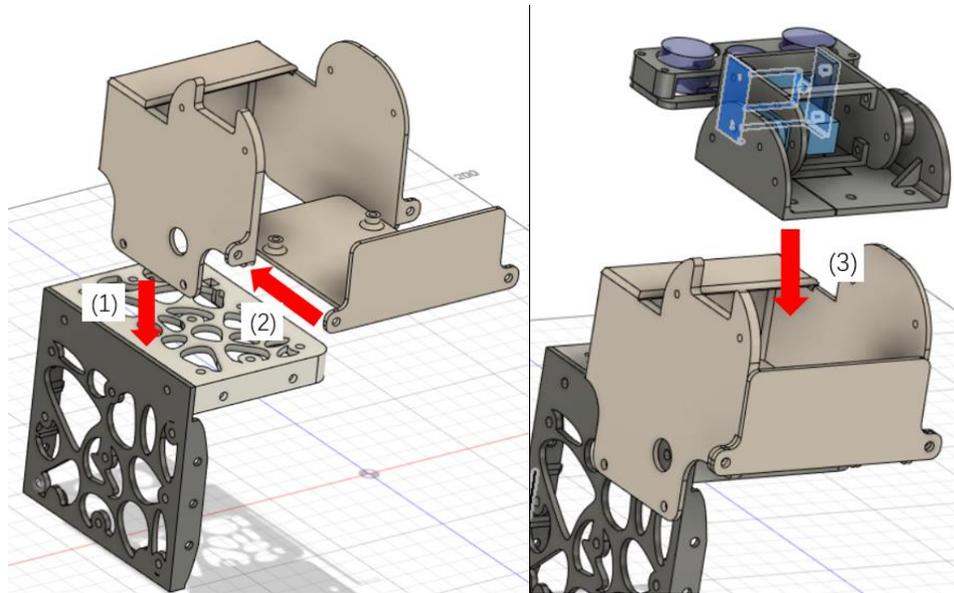


Figure 5 Installation process of the base

3) Operation

The controller provides the interaction between the user and the robot. There are 3 modes and, in each mode, those 6 buttons have their own functions. We set Mode2 to control the servos. Button1 and 2 are to control the servo above the base. Button3 and 4 are to control the servo connected to the camera. Each time you press Button1 or 3, the corresponding servo will rotate 30° . When the degree is 90° or 180° , it will rotate in the opposite direction. When you press Button 2 or 4, the corresponding servo will turn to a fixed angle. We set their angles as 135° and 45° respectively.

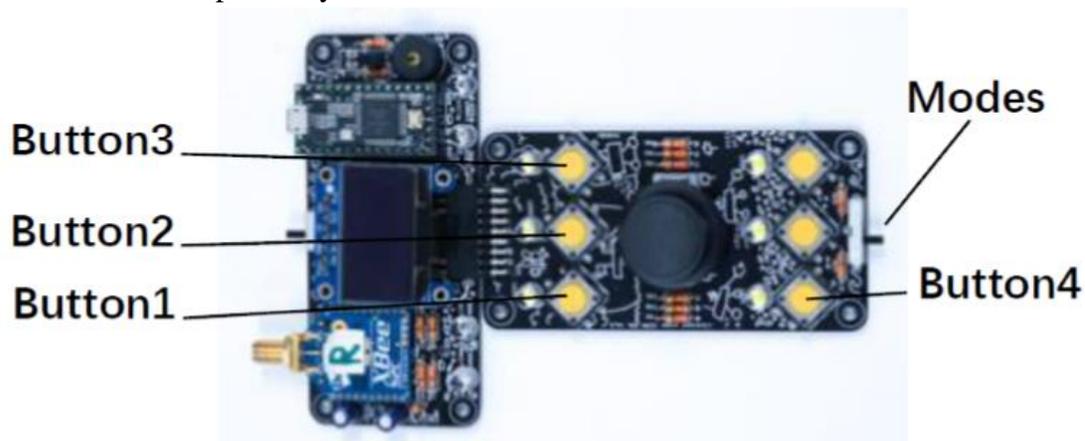


Figure 6 Controller of Bowie

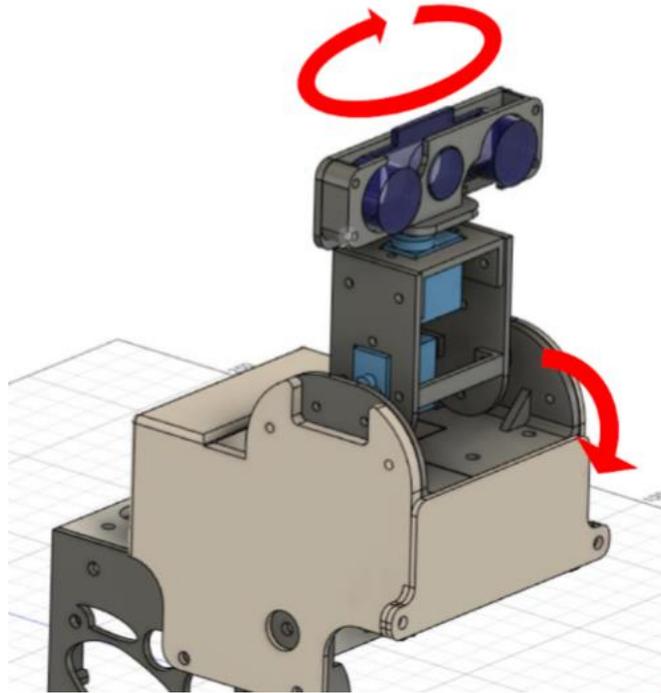


Figure 7 How the camera moves

4) Maintenance

The thickness of the 3D printed components is 2~3 mm, which is strong enough to face the impact. When some of them are broken, it will be fast and easy to make new parts by 3D printer to replace them. Some electric components are exposed in the air directly, so it is very important to protect the device from water.

2.1.3 Detailed Instructions for Object Detection

Our robot ‘Bowie’ is a beach cleaning robot and could work manually and automatically. Under automatic mode, ‘Bowie’ can recognize the garbage, move to and pick up it. However, the current version has a defect. ‘Bowie’ is not able to work at night since the limit of camera and detection method. Thus, according to our clients’ requirement, i.e. making ‘Bowie’ can work at night, we replaced the camera with IR camera to obtain vision and modify the code to make the garbage in the IR image could be detected.

In order to update the function of ‘Bowie’, we need to finish two steps. First, we need to install TensorFlow dependencies on Raspberry Pi and then upload new object detection code to Raspberry Pi.

- 1) Install TensorFlow on raspberry pi

Here we found a tutorial post on GitHub: https://github.com/EdgeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md.^[1] Follow the first section to help you set up TensorFlow Lite on Raspberry Pi and ready for object detection. All steps are listed here:

- a. Update the Raspberry Pi
 - b. Download relative repository and create virtual environment
 - c. Install TensorFlow
- 2) Update the raspberry pi

Firstly, open a terminal and issue to update Raspberry:

```
sudo apt-get update  
sudo apt-get dist-upgrade
```

While updating we make sure the camera interface is enabled in the Raspberry Pi Configuration menu. Click the Pi icon in the top left corner of the screen, select Preferences -> Raspberry Pi Configuration, and go to the Interfaces tab and verify Camera is set to Enabled. If it isn't, enable it now, and reboot the Raspberry Pi.

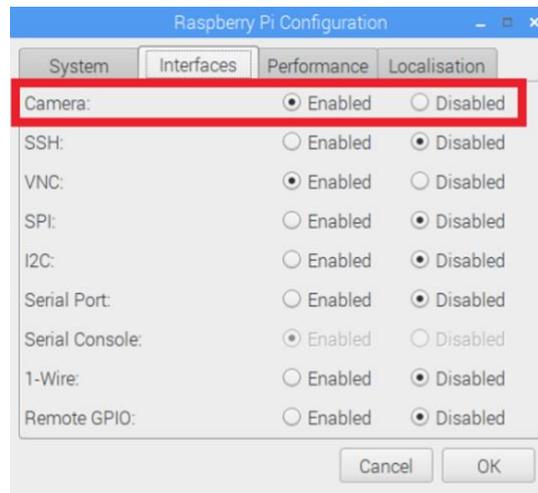


Figure 8 Enable the camera

- 3) Download relative repository and create virtual environment

Next, clone this GitHub repository by issuing the following command. The repository contains the scripts we'll use to run TensorFlow Lite, as well as a shell script that will make installing everything easier.

```
git clone https://github.com/EdgeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

This downloads everything into a folder called TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi. Rename the folder to "tflite1" and then cd into it:

```
mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1  
cd tflite1
```

Next is to create a virtual environment called "tflite1-env" by issuing:

```
sudo pip3 install virtualenv
```

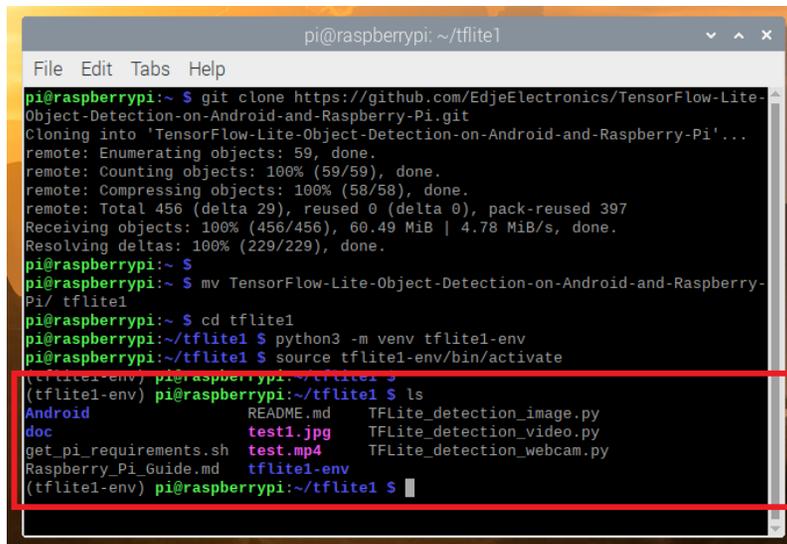
And:

```
python3 -m venv tflite1-env
```

This will create a folder called tflite1-env inside the tflite1 directory. The tflite1-env folder will hold all the package libraries for this environment. Next, activate the environment by issuing:

```
source tflite1-env/bin/activate
```

At this point, here's what your tflite1 directory should look like:



```
pi@raspberrypi: ~/tflite1
File Edit Tabs Help
pi@raspberrypi:~ $ git clone https://github.com/EdgeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
Cloning into 'TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 456 (delta 29), reused 0 (delta 0), pack-reused 397
Receiving objects: 100% (456/456), 60.49 MiB | 4.78 MiB/s, done.
Resolving deltas: 100% (229/229), done.
pi@raspberrypi:~ $
pi@raspberrypi:~ $ mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/ tflite1
pi@raspberrypi:~ $ cd tflite1
pi@raspberrypi:~/tflite1 $ python3 -m venv tflite1-env
pi@raspberrypi:~/tflite1 $ source tflite1-env/bin/activate
(tflite1-env) pi@raspberrypi:~/tflite1 $
(tflite1-env) pi@raspberrypi:~/tflite1 $ ls
Android          README.md       TFLite_detection_image.py
doc              test1.jpg       TFLite_detection_video.py
get_pi_requirements.sh  test.mp4       TFLite_detection_webcam.py
Raspberry_Pi_Guide.md  tflite1-env
(tflite1-env) pi@raspberrypi:~/tflite1 $
```

Figure 9 tflite1 directory

4) Install TensorFlow

Using a shell script wrote by author that will automatically download and install all the packages and dependencies. Run it by issuing:

```
bash get_pi_requirements.sh
```

Here we have our TensorFlow ready on Raspberry Pi.

5) TensorFlow Object Detection API

When our TensorFlow has set up on Pi, then we could upload the code of object detection to it. The code we use is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. Here is the link of TensorFlow object detection API: <https://learn.adafruit.com/tensorflow-in-your-browser-object-detection-with-bounding-boxes/opening-the-notebook-2>.^[2]

The first part of our code is downloading the model data which could help us get the memory of a pre-trained model.

```

!git clone https://github.com/tensorflow/models
checkpoint_name = 'ssdlite_mobilenet_v2_coco_2018_05_09'
!wget http://download.tensorflow.org/models/object_detection/{checkpoint_name}.tar.gz
!tar -xf {checkpoint_name}.tar.gz
checkpoint = '{0}.ckpt'.format(checkpoint_name)
!cd /content/models/research && protoc object_detection/protos/*.proto --python_out=.
print('Setup successful!')

```

Figure 10 Download the model

Here you need to make sure that your version of TensorFlow is 1.x. You can run this cell to set it up:

```
[2] %tensorflow_version 1.x
```

The second part is about configuring TensorFlow. These steps start TensorFlow and read the downloaded model data into memory so we can use them.

```

[2] # NOTE: Pieces taken from https://github.com/tensorflow/models/blob/master/research/ob
# The github.com/tensorflow/models is distributed under the Apache 2.0 license.

import numpy
import tensorflow as tf

tf.reset_default_graph()
!rm -f /content/logs/*

with tf.get_default_graph().as_default() as graph:
    summary_writer = tf.summary.FileWriter("/content/logs/", flush_secs=1)

# This section builds a "graph" in TensorFlow to explain how to process the data.
jpeg_input_tensor = tf.placeholder(tf.string, ()) # We will provide a JPEG to TF.

# First, instruct TF to decode the JPEG string into a matrix.
image = tf.image.decode_image(jpeg_input_tensor)

```

Figure 11 TensorFlow configuration

Then comes to the final part. This section creates the video input element and connects it to TensorFlow.

```

import base64
import html
import io
import time

# setup path so that mobilenet_v2 can be found.
import sys
sys.path.append('/content/models/research')
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils

# Taken from https://colab.research.google.com/notebooks/snippet
from IPython.display import display, Javascript
from google.colab.output import eval_js
import numpy
import PIL.Image

def start_input():
    js = Javascript("""
        var video;
        var div = null;
        var stream;
        var captureCanvas;
        var imgElement;
    """)

```

Figure 12 Video input element and connection to TensorFlow.

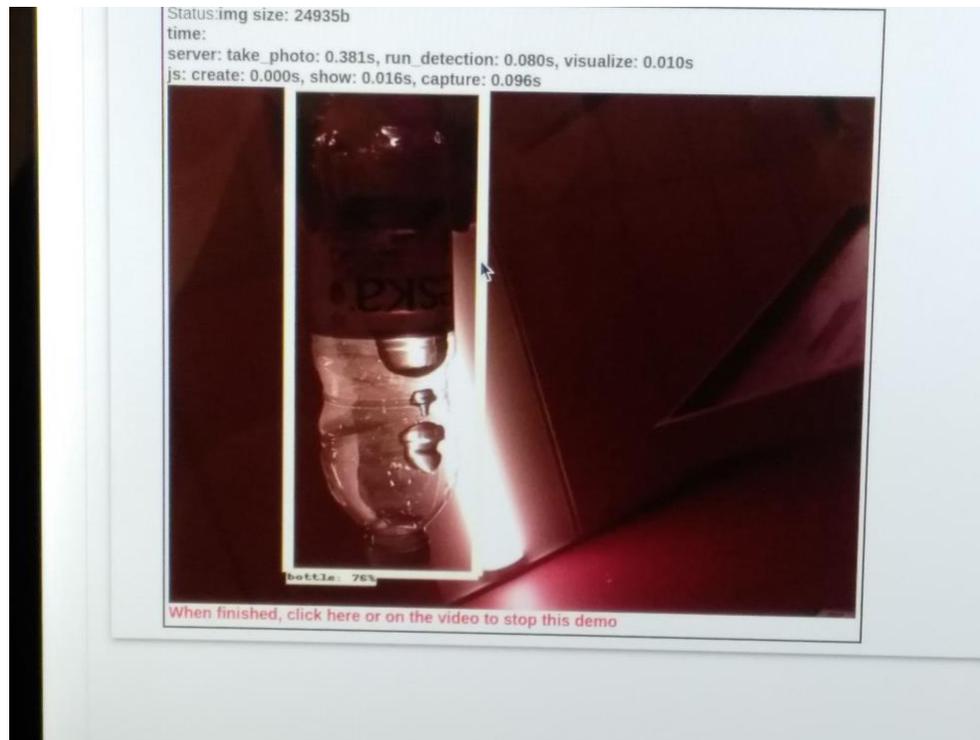


Figure 13 Object detection result example

2.1.4 Safety Guidelines

For the assembly part of the camera arm, all operations need to be performed with the power off.

2.1.5 Troubleshooting and Technical Instruction

1) Camera arm problem in supervised working situation

If the controller cannot control the camera's angle normally, first adjust the mode of the controller to mode1 or mode3, and then observe whether the controller can control other behaviors of the robot normally. If other functions can be normally performed by the controller, the controller problem is eliminated. If other functions are also not performed in the right way, then the problem is on the controller or the raspberry pi.

If other mode can work normally, then you should connect the camera arm to the test program (SweepTest.ino), a simple loop function, observe whether the 2 degrees of freedom of the camera arm can rotate normally. If not, the cause of this problem is the failure of the servo, and the problem can be solved by directly replacing the servo. If the camera arm can work normally at this time, the cause of the problem is a malfunction of the raspberry pi connected to the controller and the camera arm.

2) Camera arm problem in unsupervised working situation

If the camera arm cannot rotate normally during the robot's automatic operation, first you need to connect the camera arm to the test program (SweepTest.ino), a simple loop function, and observe whether the 2 degrees of freedom of the camera arm can rotate normally. If not, the cause of this problem is the failure of the servo, and the problem can be solved by directly replacing the servo. If the camera arm can work normally at this time, the cause of the problem is a malfunction of the raspberry pi connected to the controller and the camera arm.

3) Object detection problem

If the provided object detection model does not work properly, you should first replace the model with a sample model, the tflite model (coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip), and then test whether the tflite model works properly. If it does not work, the problem lies in the hardware, which is raspberry pi. If the tflite model can work normally, it is the problem of the model imported.

4) TensorFlow installation problem

If the object detection model of the final prototype cannot be used normally, and the hardware damage has been eliminated, then the problem lies in the installation of TensorFlow. Following the provided tutorial, first of all, you need to check whether the raspberry pi used has been updated, whether the virtual environment has been successfully built, whether the dependencies have been downloaded in the correct path, and finally, whether the model used has been installed in the correct folder.

2.2 Design File

MakerRepo link

<https://makerepo.com/jessical022/environmental-robot>

Including:

- 1) coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip (testing object detection model, tflite)
- 2) controller.zip (the Arduino code for the controller)
- 3) Copy_of_Mobilenetv2+_SSDLite.ipynb (code for pre-trained object detection model)
- 4) SweepTest.ino (camera arm testing code)
- 5) parts.rar (3D printing files of all parts of the camera arm, and the structure of the whole camera arm)

TensorFlow installation on raspberry pi tutorial [1]

https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md.

3 Conclusions and Recommendations for Future Work

The environmental robot is Low-Cost robotics for environmental applications. This robot is particularly designed for beach cleaning missions. It can recognize several types of garbage and then shovel it to its container.

In order to improve Bowie's working efficiency and reduce its control difficulty, we mainly complete two design improvements in this project. The first is about the camera arm, and the other is about improvements in the object detection method. With regard to the camera arm, we provide the camera with two degrees of rotation freedom that can be controlled by the controller. Regarding the improvement of automatic object detection, we introduce a new TensorFlow machine learning model based on IR images, which can provide near real-time automatic detection effects. Although the recognition accuracy of different objects is not the same, the overall accuracy is stable at about 80%.

This report presents the important features of the final prototype, and then provide the users with instructions on how the prototype was made, how it works, how to install and operate it as well as maintenance instructions. The troubleshooting part explains the problems that may be encountered, and also presents the follow-up instructions.

The size of the training data set and the proportion of various kinds of pictures can greatly affect the accuracy of the model. In future work, our plan is to find a better pre-trained model with proper dataset, or we can train the model based on our own dataset if we have strong hardware support.

4 Bibliography

[1] https://github.com/EdgeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md.

[2] <https://learn.adafruit.com/tensorflow-in-your-browser-object-detection-with-bounding-boxes/opening-the-notebook-2>

APPENDICES

Work completion record:

<https://www.robotmissions.org/category/news/collab-logs/>