SOFE 4590U: Embedded Systems
Fall 2021
Assignment 2 : Performance evaluation of different architectures on QEMU

Owais Quadri
100697281
Due Date: Nov 11, 2021

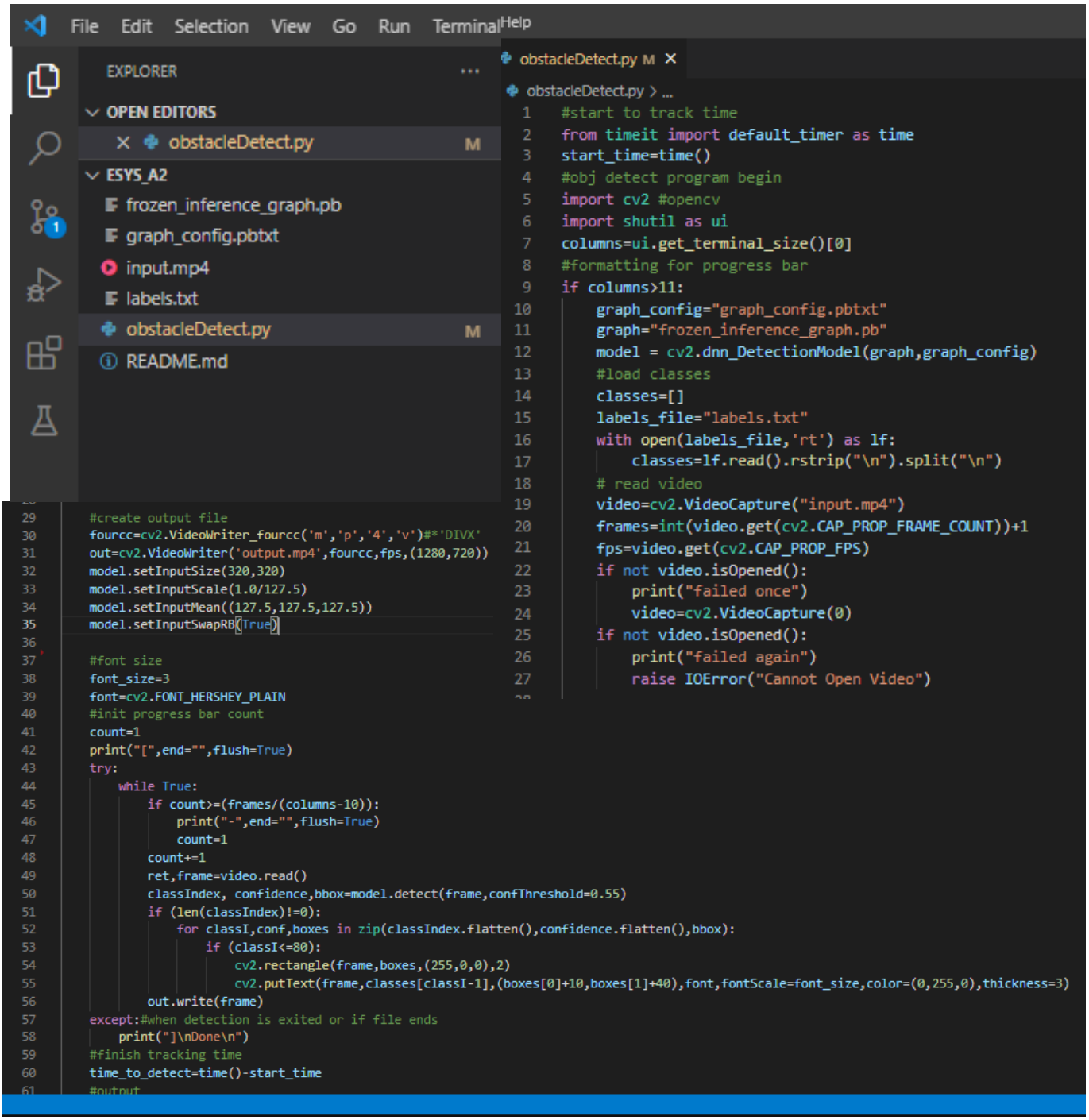
Google Drive Video Link:
https://drive.google.com/file/d/1fIxSsxy0OFSN61zIBJGbnWo7Bw_JMa3F/view?usp=sharing

Introduction

Image processing applications can often have timing constraints. For example, consider the computer vision system of an autonomous vehicle when an object is in the path of the moving vehicle, we require that objects to be detected. The goal therefore of this assignment is to develop a new or reuse an existing (please refer to the source) computer vision application using any programming language and run on QEMU for two different architectures to analyze the time difference it takes for all the objects detected between two different architectures. You can either generate videos on your own or obtain them from online sources (please refer to the source).

Deliverables:

1. Code Overview

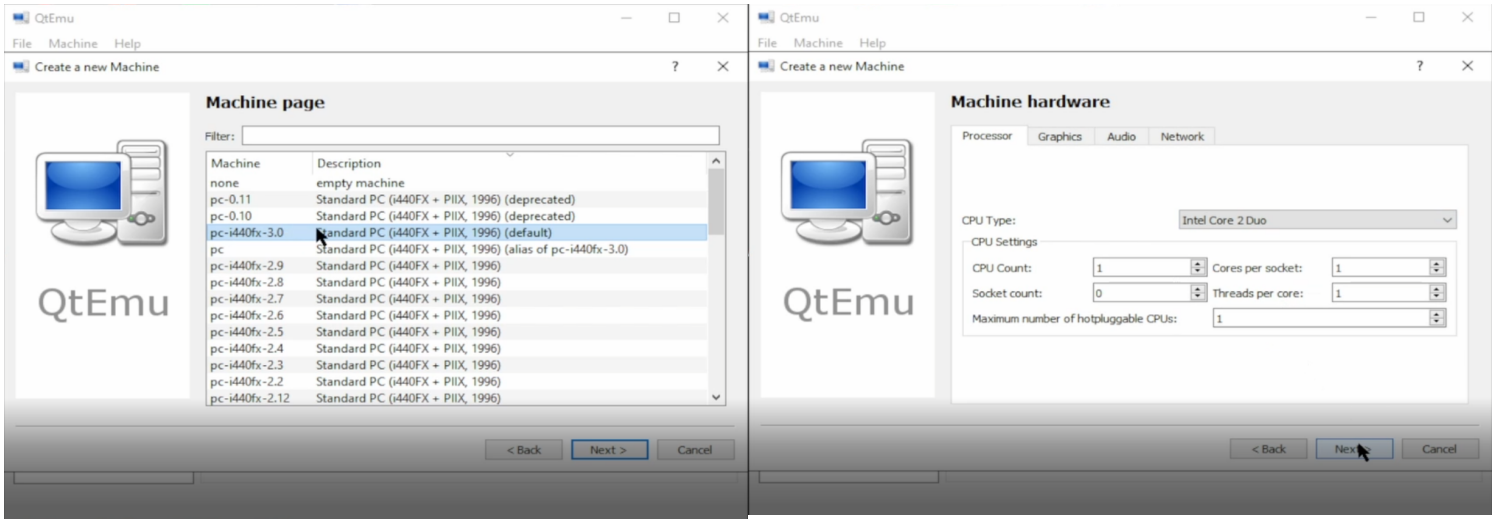


The code for obstacle detection was coded by me, after watching a guide on youtube for opencv[1]. I created the code before testing the code on the different architectures because I wanted to make sure that the code would run properly. I used the timeit.default_timer to measure the time that it took (performance) to process the video. I also included feedback to show how long that the processing should take (the progress line proceeds until near the end of the terminal window horizontally). I used shutil to find the terminal width and proceeded with that. Then, I loaded the tensorflow model[2] using the graph and config files[3] using opencv (cv2). After that, I loaded the labels[4] for the model, I loaded the input.mp4[5] video and while the video was being iterated through, each frame was getting its objects detected and rectangles drawn around each detected item with its corresponding labels. Then, the output video writer was being written to and when the loop is broken (video ends or is escaped by using 'Ctrl+C'), the time for processing is calculated, the result is printed and the processed video is written to a file called 'output.mp4'.

2. Snapshots of obstacle detection running on different architectures:

First Architecture: Machine = pc-i440fx-3.0 : PC(i440FX + PIIX, 1996) ,

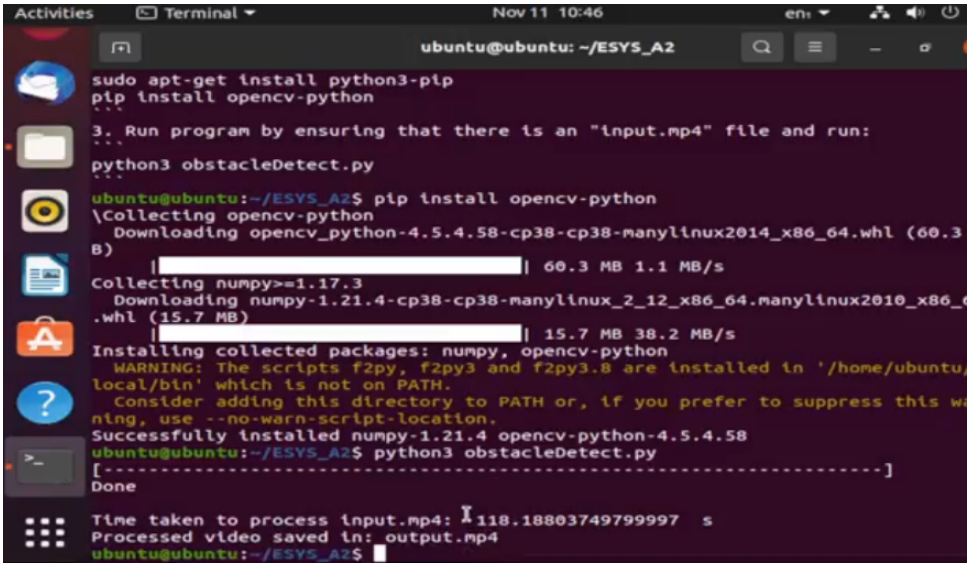
CPU Type = Intel Core 2 Duo



Program running on first architecture:

- Input video screenshot:

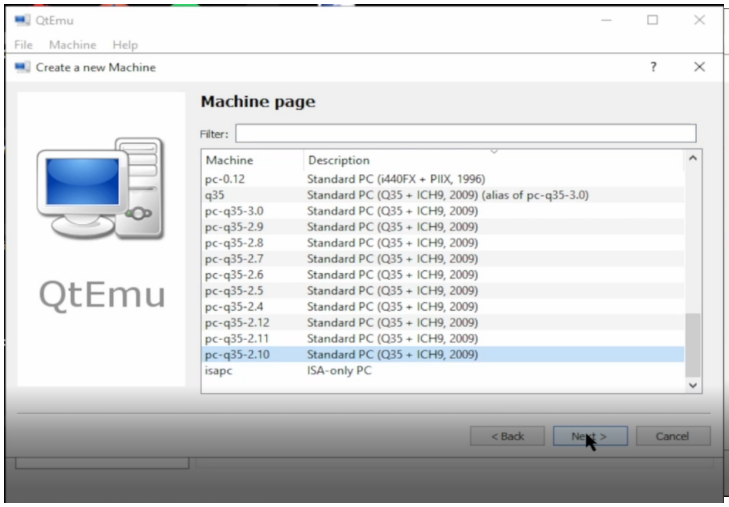
- Terminal :

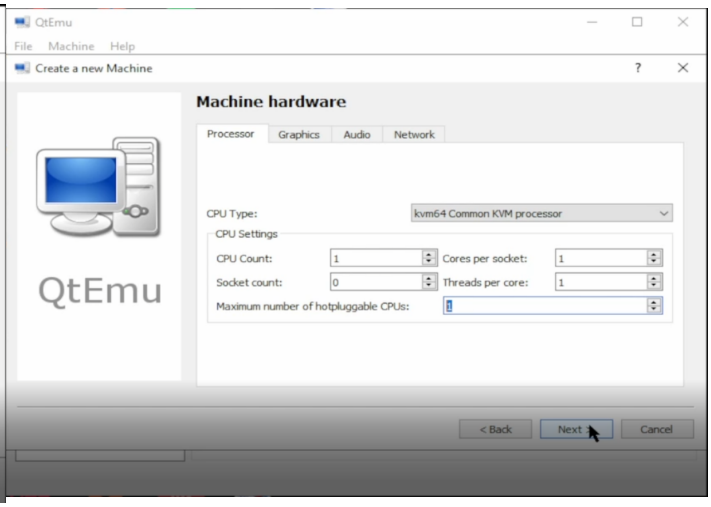


- Output video screenshot:

Second Architecture: Machine = pc-q35-2.10 : PC(Q35 + ICH9, 2009) ,

CPU Type = kvm64 Common KVM processor

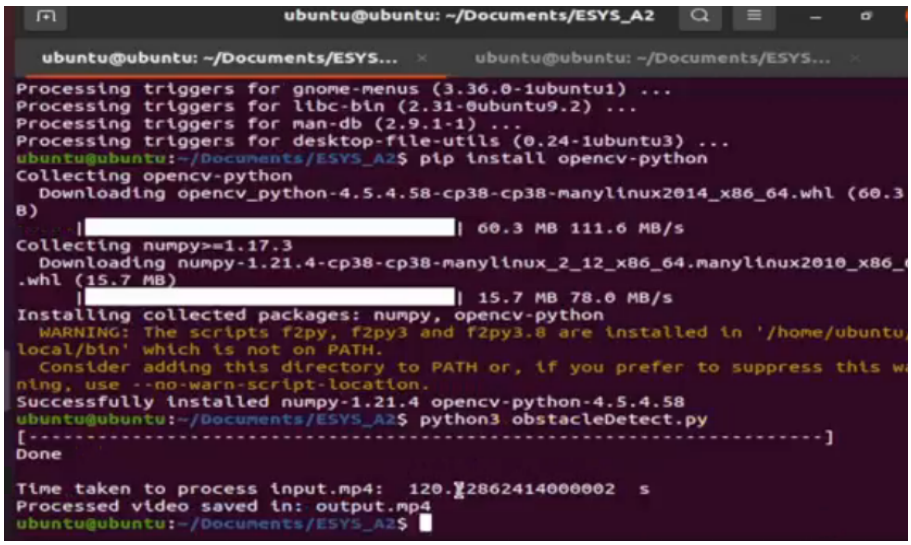




Program running on second architecture:

- Input video screenshot:

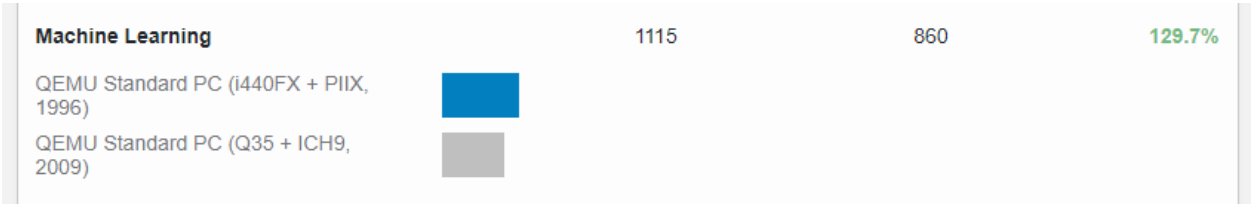
- Terminal :



- Output video screenshot:

Explanation for differences:

In some cases, the Q35 is better than the i440FX , but for machine learning applications, this source indicates that the i440FX should perform 29.7% better than the Q35 [6]. Meanwhile, when tested, the i440FX completed the processing for a 60 second video in 118.1880 seconds where the Q35 completed the processing on the same input video in 120.2286 seconds. This means that the i440FX ran only 1.73% faster than the Q35 machine.



3. Link to Video:
https://drive.google.com/file/d/1fIxSsxy0OFSN61zIBJGbnWo7Bw_JMa3F/view?usp=sharing

References

[1] OpenCV Guide https://www.youtube.com/watch?v=RFqvTmEFtOE

[2] Tensorflow model used:
https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API

[3] Config file: https://gist.github.com/dkurt/54a8e8b51beb3bd3f770b79e56927bd7

[4] Classes label list data: https://github.com/pjreddie/darknet/blob/master/data/coco.names

[5] Video used: https://www.youtube.com/watch?v=aMJNBUDk69Q (2:23-3:23)

[6] i440FX vs Q35 https://browser.geekbench.com/v5/cpu/compare/4589842?baseline=4589992