Smart Home with Computer Vision: Object Detection with Pre-trained MobileNet SSD

Jiawen Xu
Electrical and Computer Engineering
jix034@ucsd.edu

Abstract

In this project, we developed a smart home program for home monitoring with application of computer vision. We utilize `mobilenet-ssd` model and OpenCV library to implement object detection and classification features from the video capture by the webcam. With the object being detected, we further predicted the current status of the objects in the current environment.
Key words: Computer Vision, IoT, Object Detection, Smart Home, GUI, MobileNet SSD

1 Introduction

1.1 Motivation

Smart home technology recently have emerged these days, and computer vision technologies such as object detection, object classification could be a potential method to bring further advancement on smart home devices. Smart home technology can bring better living quality and convenience to people and society.[4] Smart home has been used in different fields such as medical treatment, construction, and energy management. It is a new trendy way of to bring programmablility to benefit our daily life. [6]

1.2 Problems Identification

Computer vision is helpful for interaction between human and machines. [5] However, the application of computer vision on smart home devices is still undergoing due to the latency for machine learning model training and inefficient compatibility of the machine learning model to the actual product. In this project, we implemented a Graphical User Interface (GUI) for user-friendly operations on complicated deep learning model used for computer vision.

1.3 Key Parts of Approach

We mainly use Open Source Computer Vision (OpenCV) library for this project. OpenCV provides wide range of functions and algorithms for developers and researchers in computer vision. [1] We decided to use mobilenet-ssd model for object detection and classification. The reason we choose this is because it is a relatively light weight which is better for performance. For model training, we uses pre-trained weight of the mobilenet-ssd model as this project is more focus on application.[8]

1.4 Result Summary

In our project, we successfully implemented the object detection and classification with relatively high accuracy. We then made judgements of the current status of the environment based on what we detected from our camera. We successfully set up communication between two hosts for video

transferring and message transferring using socket programming. We designed a user-friendly GUI for all the features mentioned above.

2 Related Work

Patchava (et al) presents an example of utilizing computer vision and Raspberry Pi to detect the status of objects in Smart Home. [3] In their work, each object is monitored and detected by Raspberry Pi with data updated to database along with live streaming from the cameras. However, the system implemented relies on sensors to detect the status of each object. In our project, we will try to improve this design by reducing the use of sensors and applying object detection technique. Similar to Patchava (et al)'s work, we will implement a graphical user interface to visualize our data and result. However, based on time constraint, we may not choose to implement it with database server. Instead, we will use simple socket programming for local communication between Raspberry Pi and user's terminal.

Younis et al developed object detection utilizing pretrained`mobilenet-ssd` model. In their work, they identify the object with the deep learning pre-trained model MobileNet for Single Shot Multi-Box Detector (SSD). The Mobilenet has trained millions of images. [8] Our project is also similar to their work. After we implemented the object detection and classification, we also designed user interface and network communication features on top of that.

Suryatali develops object detection using OpenCV library in Embedded Linux system as well as using Raspberry Pi. [7] Comparing to Patchava (et al)'s work which uses SimpleCV, using OpenCV is beneficial for data visualization along with other features. SimpleCV is considered as a lightweight library for computer vision, whose features are limited, while OpenCV has a more complex library.

3 Method

3.1 Overall Structure

Figure 1 illustrates the overall structure of our final project. The webcam takes a video and send the video stream as frames to the pre-trained mobilenet. The MobileNet will predict the object types and position in the forms of bounding boxes and class type. The class types in our dataset includes up to 91 types. We are using coco dataset for the model. After the MobileNet predicted the positions and the class types, we will add the bounding box to the original videos and either display it locally or send it to our remote user. Our remote user will construct a server-client model for data transmission with Python built-in socket structures. The remote user will then display the frames received from our smart home remotely.

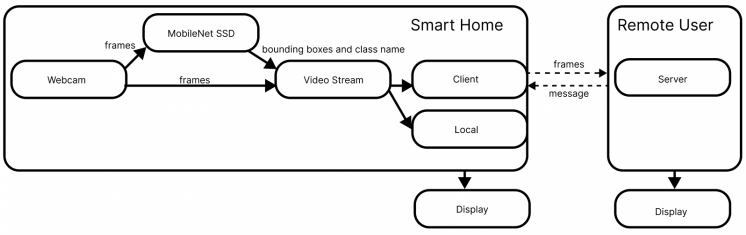

Figure 1: Overview of our project

3.2 Dataset

The dataset we are using is COCO (Microsoft Common Objects in Context) dataset. COCO dataset is used for large-scale object detection, segmentation, and captioning. It includes up to 91 stuff catgories for object detection. We are using the 2017 release version.

3.3 MobilenetSSD Model

In our project, we utilize the deep learning pre-trained model MobileNet for Single Shot Multi-Box Detector (SSD). MobiltNet SSD is a popular deep learning model for real-time object detection in images and videos. The MobileNet SSD combines the MobileNet architecture for efficient feature extraction with the SSD framework for accurate object localization and classification. It is good for real time detection on webcam where multiple objects will be detected with the data stream. Compared to YOLO (You Only Look Once) model, MobileNet SSD is relatively light weight. Mobilenet SSD will output the bounding box and object class from the input image. The SSD object detection model will use Mobilenet for better object detection for webcam.

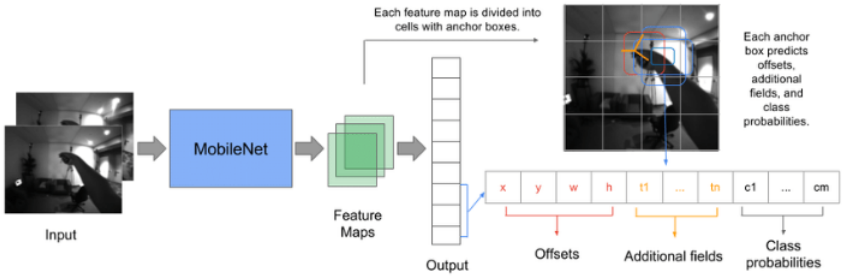

Figure 2: Structure of MobilenetSSD [2]

3.4 Image Processing and Object Detection

In this project, the videos is captured by our webcam. The OpenCV library is used to process the videos into frames. Each frame will be fed into the trained MobilenetSSD Model, which outpus the bounding box and the class type. The class type and the bounding box will then be added into the frames using OpenCV. Figure 3 shows our output of object detection and classification.

Figure 3: Object Detection and Classfication Demo

3.5 Movement Detection

Our movement detection algorithm is based on the relative change of the position of boxes between each frame. The algorithm for movement detection can roughly predict if the object is moving by calculating the relative change of the bounding box for each object with regard to the steady state position.

Our algorithm can be expressed mathematically as follow:
A bounding box can be defined as

$$B = \{(x_1, y_1), (x_2, y_2)\}$$

where (x_1, y_1) represents the left upper position and (x_2, y_2) represents the right lower position of the bounding box.
Define $\Delta(t) = \{(\Delta x_1(t), \Delta y_1(t)), (\Delta x_2(t), \Delta y_2(t)\}$ to be the change of a bounding box at time t, we have the following relationship:

$$B(t) = B(t-1) + \Delta(t)$$

$$\text{moving} = \begin{cases} \text{True} & \text{if } \frac{(\sum \Delta(t))^2}{(x_2(t)-x_1(t))(y_2(t)-y_1(t))} > \lambda \\ \text{False} & \text{otherwise} \end{cases}$$

where Δ is the change between each frame, A is the area of the bounding box, and λ is the threshold.

Figure 5 shows our result and output generated on both terminal and our designed GUI. I was in fact moving the bottle in the camera, and body was slightly shaking when I was doing that. My algorithm successfully detected it and displayed it on both terminal and the GUI.

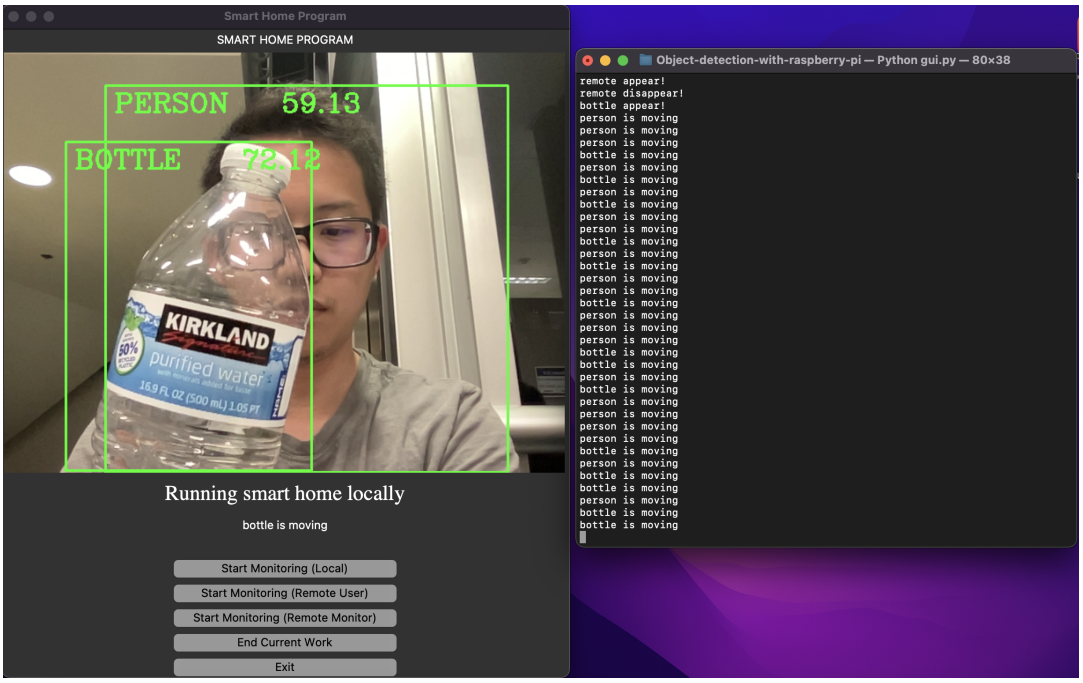

Figure 4: Motion Detection Demo

3.6 Network Communication Interface

Our remote user and remote monitor features are programmed by socket library in python. For each frame to be sent, the sender (monitor) will calculate the length of the image frame and send it first before sending the frame. After the user reading the length of the frame, it will read just enough bytes for that frame. The data sent right after will be the length of the message packet which includes the movement data. Then the monitor will send the movement data as a packet, and the remote user will

read just enough bytes for the user data. Every time the monitor send a image packet or a packet, it will wait for the response from the receiver to make sure that it is ready to receiver the next message. Figure 6 shows the example of running remote user and remote monitor on two GUI with our network communication interface.

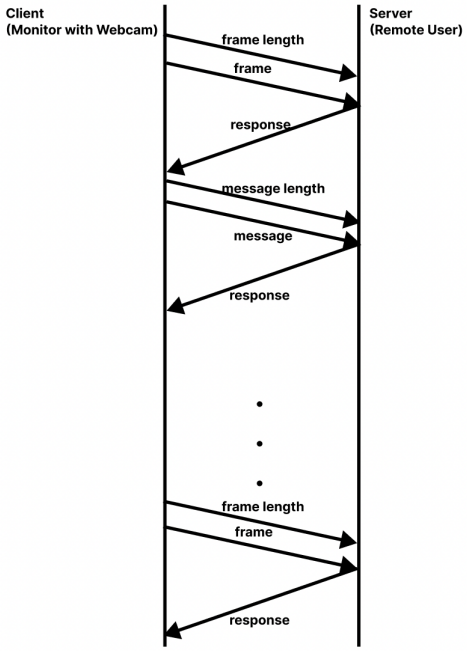

Figure 5: Network Setup

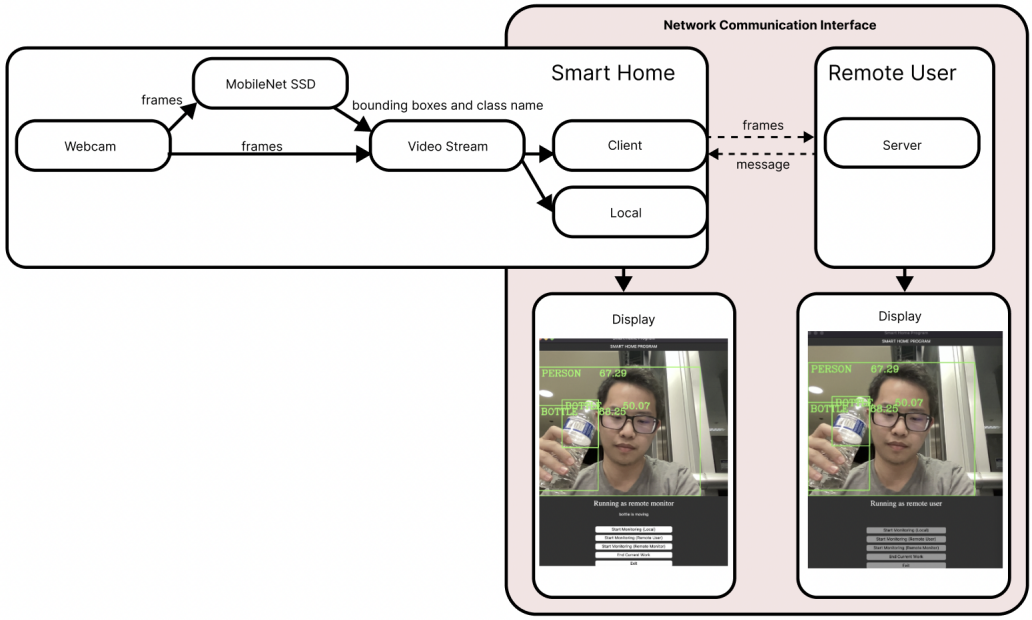

Figure 6: Illustration of network communication interface

3.7 Graphical User Interface (GUI)

We use tkinter library in python for our graphical user interface. The tkinter library provides us a way to create visualize options for user. In our main GUI, we provides 5 options. If we select "Start Monitoring (Local), then no network connection will be built. The program will utilize the

default camera to capture and process the video frames and display it on the GUI. If we select "Start Monitoring (Remote User)", then it will enter the user mode and wait for the image and packet send from the remote monitor. If we select "Start Monitoring (Remote Monitor"), it will connect to the remote user and then start the object detection program. If we select the "End Current work", it will exit the current job and go back to the welcome page. If we select the "exit", the program will exit. This GUI does not provide a way to configure the IP address. The user may need to configure the address in the source code. We are using 127.0.0.1 by default for localhost.

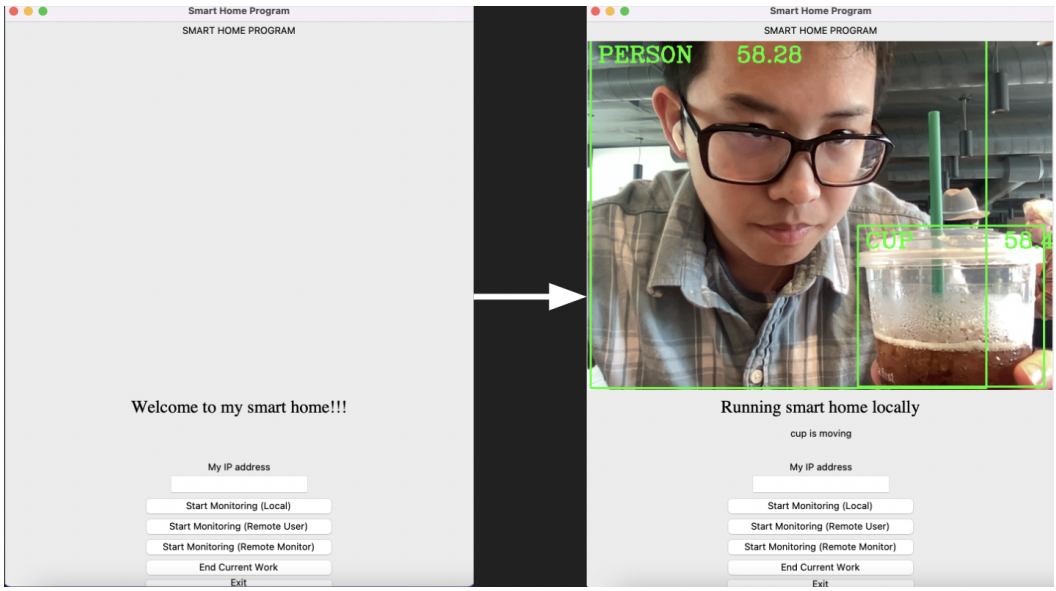

Figure 7: Main GUI

4 Experiments

4.1 Experimental Setup (Dataset)

The model is trained with the COCO dataset. There are 91 classes in the coco data set that is used for training the MobileNet SSD. In our experiment, we will select 10 class that are most commonly seen at a household for experiment.

4.2 Object Detection and Classification

For evaluation of object prediction, we collected 14 objects for experiments and we will measure the accuracy at different brightness, 10 of which are bottles while the rest of 4 are objects that are similar to bottles. We define the metrics as follows:

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions made}} * 100\%$$

$$\text{false positive} = \frac{\text{incorrect predictions}}{\text{total predictions made}} * 100\%$$

$$\text{miss rate} = \frac{\text{unboxed objects}}{\text{total objects in the set}} * 100\%$$

The best case is having high accuracy, lower false positive, and lower miss.

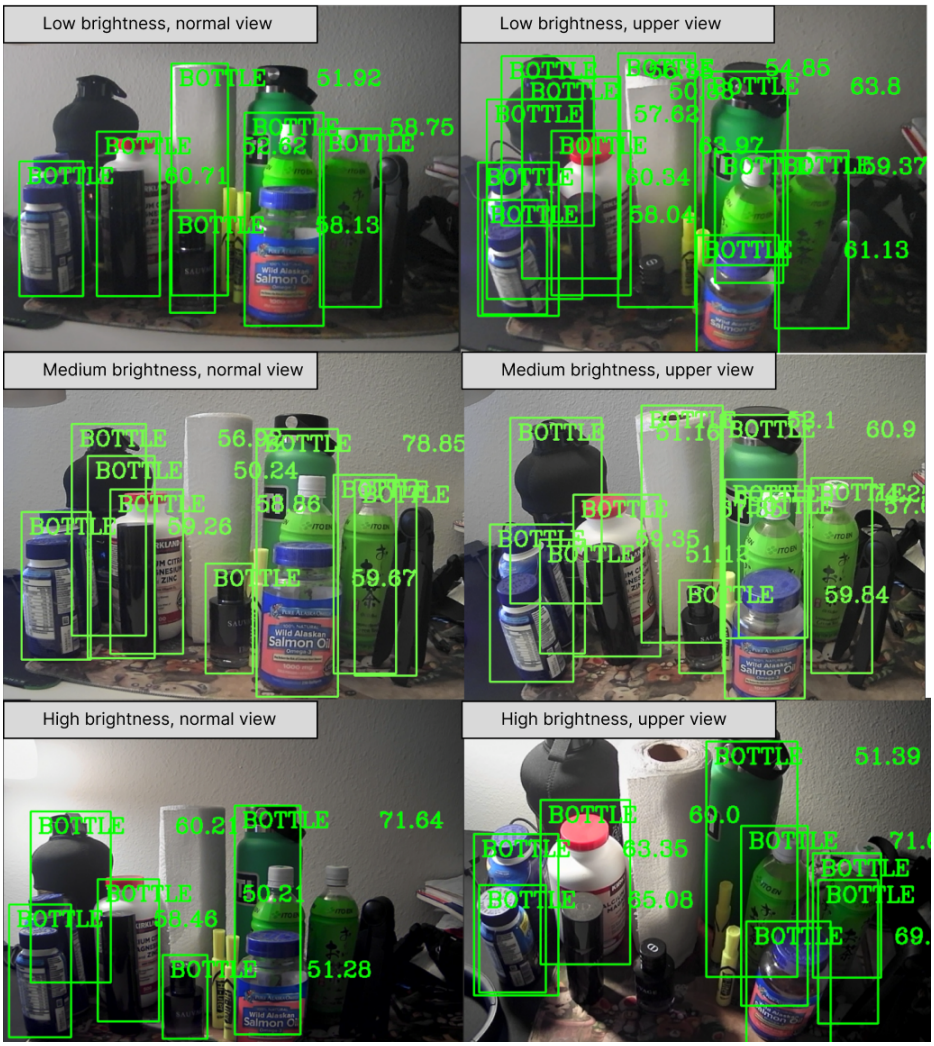

Figure 8: Result for Experiment 1

(brightness, view)	weak,normal	weak,upper	medium,normal	medium,upper	high,normal	high,upper
Accuracy	83.3%	72.7%	75%	70%	100%	100%
False Positive	16.7%	27.3	25%	30%	0%	0%
Miss Rate	57.1%	35.7%	28.6%	21.4%	42.9%	42.9%

From our test results, we can see our object detection has high accuracy with highest brightness at either perspective of view (normal or upper), and low accuracy at medium brightness. However, the miss rate is lowest with medium brightness compared to other environments. We can see that brightness and perspective is relevant to the performance of our model. Also, our model has a relatively sensitive algorithm for object detection as well as the high performance that works for real time image processing.

4.3 Object Motion Identification

In this experiment, we will test the object motion with myself finishing 4 types of movement to evaluate our object motion detection. Our motion has 4 types: appear, disappear, moving, non-moving. Appear and disappear are relatively easy to identify, so we only test the moving and non-moving cases.

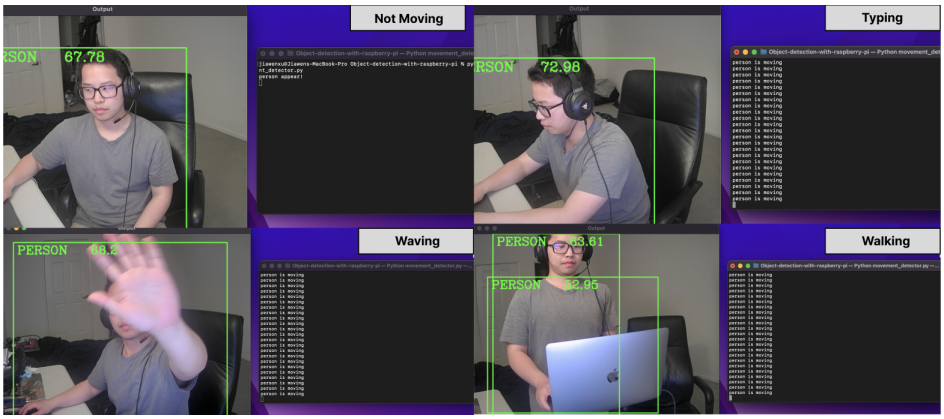

Figure 9: Result for Experiment 2

	not moving	typing keyboards	waving hands	walking
Result	PASS	PASS	PASS	PASS

I test our motion detection by finishing 4 movements. 3 of them are actual movement while 1 of them is just staying still. Our motion detection algorithm detected what I did based on the relative change of the boxes among each frame.

5 Supplementary Material

`https://youtu.be/rJOkTSjSeU8`

References

[1] Gary Bradski, Adrian Kaehler, et al. Opencv. *Dr. Dobb's journal of software tools*, 3(2), 2000.

[2] Anurag Singh Choudhary. Object detection using yolo and mobilenet ssd, Sep 2022.

[3] Vamsikrishna Patchava, Hari Babu Kandala, and P Ravi Babu. A smart home automation technique with raspberry pi using iot. In *2015 International Conference on Smart Sensors and Systems (IC-SSS)*, pages 1–4, 2015.

[4] Rosslin John Robles and Tai-hoon Kim. Applications, systems and methods in smart home technology: A. *Int. Journal of Advanced Science And Technology*, 15:37–48, 2010.

[5] V Suma. Computer vision for human-machine interaction-review. *Journal of trends in Computer Science and Smart technology (TCSST)*, 1(02):131–139, 2019.

[6] Shruthi Suresh and P V Sruthi. A review on smart home technology. In *2015 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–3, 2015.

[7] Abhijeet Suryatali and V. B. Dharmadhikari. Computer vision based vehicle detection for toll collection system using embedded linux. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pages 1–7, 2015.

[8] Ayesha Younis, Li Shixin, Shelembi Jn, and Zhang Hai. Real-time object detection using pre-trained deep learning models mobilenet-ssd. In *Proceedings of 2020 the 6th international conference on computing and data engineering*, pages 44–48, 2020.