

國立成功大學工業物聯網期末專案

影像辨識居家即時防盜系統

Anti-theft system using embedded IIOT and TensorFlow
lite

第十一組

利彥儒 廖沁旋

中華民國 111 年 01 月

January 2022

一、	摘要	3
1.1	前言	3
1.2	研究動機與目的	3
二、	系統架構.....	4
2.1	裝置架構.....	4
2.2	硬體設施.....	4
2.3	軟體編程.....	5
2.4	運作說明.....	7
三、	參考資料.....	7

一、摘要

1.1 前言

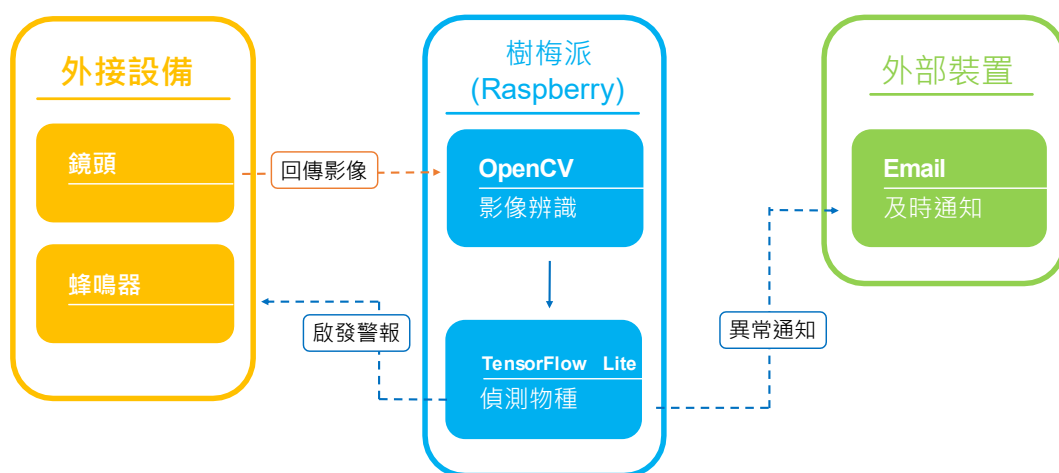
2021 年內政部統計通報指出 2020 上半年全般刑案發生數中，以竊盜案件的 3 萬 7042 件做為位居第三的一般犯罪種類[1]。其中，檢察機關指出於 100 至 109 的十年間，竊盜案先減後增，並且以 109 年 5 萬 3,624 人最多[2]。無論在台灣何處，頻繁發生的盜竊案件不僅危害到一般民眾的經濟利益，也造成保險公司、公家機關(如：警察局、法院)的困擾，更添增台灣社會秩序的不安定。

1.2 研究動機與目的


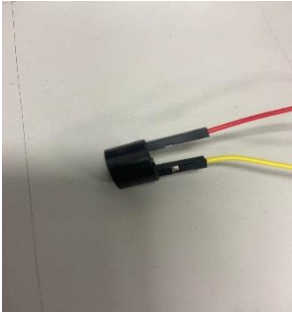
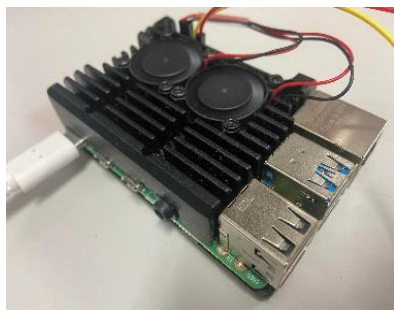
雖竊盜案在刑法中早已有相對的法規去規範，但是竊盜案件仍舊層出不窮，無論是重大竊盜、普通竊盜、交通工具竊盜……等的案件仍每日都在發生，因此本次專題我們希望可以結合樹梅派、影像辨識以及 TensorFlow lite 去製作一居家即時防盜系統，使沒有保全系統的一般住家出門也能透過此機關有效防止宵小入侵。

二、系統架構

2.1 裝置架構



2.2 硬體設施

鏡頭	蜂鳴器	樹梅派
		

2.3 軟體編程

```
import RPi.GPIO as gpio
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util

from test import *

buzzer = 17

gpio.setmode(gpio.BCM)
gpio.setup(buzzer, gpio.OUT)

class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self, resolution=(640,480), framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3, resolution[0])
        ret = self.stream.set(4, resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

        # Variable to control when the camera is stopped
        self.stopped = False

    def start(self):
        # Start the thread that reads frames from the video stream
        Thread(target=self.update, args=()).start()
        return self

    def update(self):
        # Keep looping indefinitely until the thread is stopped
        while True:
            # If the camera is stopped, stop the thread
            if self.stopped:
                # Close camera resources
                self.stream.release()
                return

            # Otherwise, grab the next frame from the stream
            (self.grabbed, self.frame) = self.stream.read()

    def read(self):
        # Return the most recent frame
        return self.frame

    def stop(self):
        # Indicate that the camera and thread should be stopped
        self.stopped = True
```

```

# Define and parse input arguments
parser = argparse.ArgumentParser()
parser.add_argument('--model_dir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the Labelmap file, if different than Labelmap.txt',
                    default='Labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                    default=0.5)
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',
                    default='1280x720')
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                    action='store_true')

args = parser.parse_args()

MODEL_NAME = args.model_dir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)
resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
use_TPU = args.edgetpu

# Import TensorFlow libraries
# If tflite_runtime is installed, import interpreter from tflite_runtime, else import from regular tensorflow
# If using Coral Edge TPU, import the load_delegate library
pkg = importlib.util.find_spec('tflite_runtime')
if pkg:
    from tflite_runtime.interpreter import Interpreter
    if use_TPU:
        from tflite_runtime.interpreter import load_delegate
else:
    from tensorflow.lite.python.interpreter import Interpreter
    if use_TPU:
        from tensorflow.lite.python.interpreter import load_delegate

# If using Edge TPU, assign filename for Edge TPU model
if use_TPU:
    # If user has specified the name of the .tflite file, use that name, otherwise use default 'edgetpu.tflite'
    if (GRAPH_NAME == 'detect.tflite'):
        GRAPH_NAME = 'edgetpu.tflite'

# Get path to current working directory
CMD_PATH = os.getcwd()

# Path to .tflite file, which contains the model that is used for object detection
PATH_TO_CKPT = os.path.join(CMD_PATH, MODEL_NAME, GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CMD_PATH, MODEL_NAME, LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

```

```

if labels[0] == '???':
    del(labels[0])

# Load the Tensorflow Lite model.
# If using Edge TPU, use special load_delegate argument
if use_TPU:
    interpreter = Interpreter(model_path=PATH_TO_CKPT,
                             experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)

interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()

# Initialize video stream
videostream = VideoStream(resolution=(imW, imH), framerate=30).start()
time.sleep(1)

#for frame1 in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
while True:

    # Start timer (for calculating frame rate)
    t1 = cv2.getTickCount()

    # Grab frame from video stream
    frame1 = videostream.read()

    # Acquire frame and resize to expected shape [1xHxWx3]
    frame = frame1.copy()
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)

    # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

    # Perform the actual detection by running the model with the image as input
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()

```

```

boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects
classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
#num = interpreter.get_tensor(output_details[3]['index'])[0] # Total number of detected objects (inaccurate and not needed)

# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
        ymin = int(max(1, (boxes[i][0] * imH)))
        xmin = int(max(1, (boxes[i][1] * imW)))
        ymax = int(min(imH, (boxes[i][2] * imH)))
        xmax = int(min(imW, (boxes[i][3] * imW)))
        state=0
        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

        # Draw label
        object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
        label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
        labelSize, baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
        label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
        cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseline-10), (255, 255, 255), cv2.FILLED) # Draw white box around label
        cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text
        if(int(classes[i])==0):
            #cv2.putText(frame, 'FALL', (30,200), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
            if(state==0):
                waring_message();
                print("send waring message!")
                state=1

                gpio.output(buzzer, gpio.HIGH)
            else:
                gpio.output(buzzer, gpio.LOW)
                #cv2.putText(frame, 'OK', (30,200), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,0), 2, cv2.LINE_AA)
                state=0

        # Draw framerate in corner of frame
        cv2.putText(frame, 'FPS: {0:.2f}'.format(frame_rate_calc), (30,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,0), 2, cv2.LINE_AA)

        # All the results have been drawn on the frame, so it's time to display it.
        cv2.imshow('Object detector', frame)

        # Calculate framerate
        t2 = cv2.getTickCount()
        time1 = (t2-t1)/freq
        frame_rate_calc= 1/time1

        # Press 'q' to quit
        if cv2.waitKey(1) == ord('q'):
            break

# Clean up
cv2.destroyAllWindows()
videostream.stool()

```

影像辨識部分參考 Github [3][4]，辨識方式、啟動警鈴及異常通知全為自己寫。

2.4 運作說明

鏡頭透過 openCV 在樹梅派中顯示畫面，當 TensorFlow Lite 偵測到人類時，樹梅派中的判斷式便會引起蜂鳴器發出聲響，並同時自動寄送入侵信件給指定信箱。

成果展示影片：<https://youtu.be/nJmZLz-UGCQ>

三、參考資料

1. 中華國內政部 (2021.)。110 年第 6 週內政統計通報。
https://www.moi.gov.tw/News_Content.aspx?n=9&sms=9009&s=212729
2. 竊盜罪案件統計分析 (2021)。https://reurl.cc/Qj9am9
3. Github. <https://reurl.cc/veqEdo>
4. Github <https://reurl.cc/xOGWQL>