CRONUS JOURNAL

Prepared for

Dr. Abhijit R. Asati

BITS Pilani,

Pilani Campus.

April 2019

CRONUS JOURNAL

Prepared for

Dr. Abhijit R. Asati

BITS Pilani,

Pilani Campus.

Prepared by:

Aayush Agarwal	2018A8PS0425P
Ananya Khandelwal	2018A8PS0417P
Ansh Shah	2018B5PS0917P
Atharv Arora	2018A3PS0568P

A report submitted as a supplement to the Apogee project.

Table of Contents

TEAM CRONUS JOURNAL

For this project we have mostly went through online resources like video tutorials and official documentation of libraries available for python. Below is a brief explanation of the basic concepts and resource links which are exhaustive for the imitation of the project.

LOCOMOTION:

PWM

The robot uses PWM signals to control the 8 servo and 4 DC motors. PWM (Pulse Width Modulation) a technique for getting analog results with digital means; is used for controlling the amplitude of digital signals in order to control devices and applications requiring power. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by the discrete switching as they react slower due to inertia. The PWM switching frequency has to be high enough not to affect the load, i.e. the resultant waveform perceived by the load must be as smooth as possible.

The term duty cycle describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on.

GPIO

GPIO stands for General Purpose Input Output. A GPIO is an uncommitted digital signal pin on an integrated circuit or electronic circuit board whose behaviour is controllable by user at run time.

The Raspberry Pi ha two rows of GPIO pins, which are connections between the raspberry pi and the real world. Output pins are like switches that the Raspberry Pi can turn on or off (like turning on/off a LED light). But it can also send a signal to another device. Input pins are like switches that you can turn on or off from the outside world (like an on/off light switch). But it can also be a data from a sensor, or a signal from another device.

PIGPIO

Pigpio is a Python module for the Raspberry which talks to pigpio daemon to allow control of GPIO pins. The module can run on Windows, MacOS or Linux and can control one or more Pi's. It has a series of predefined function which allow the user to read/write GPIO and set their modes. It creates and transmits perfectly timed waveforms.

OS Library

The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

L293D

The L293D is a 16-pin Motor Driver IC which can control up to two DC motors simultaneously in any direction. The L293D is designed to provide bidirectional drive currents of up to 600 mA (per channel) at voltages from 4.5 V to 36 V.

The documentation of the PiGPIO library can be found at - http://abyz.me.uk/rpi/pigpio/python.html

IMAGE PROCESSING:

OVERVIEW:

The image processing part has mostly been referred to from a YouTube channel called Sentdex which explains the common functions used by cv2 which is OpenCV's library for python. The official cv2 documentation can be found at :

TUTORIAL LINKS:

Face detection and Eye detection

https://www.youtube.com/playlist?list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq

1.In the first video, an introduction to image processing and loading images is taught.

Link:-
https://www.youtube.com/watch?v=Z78zbnLlPUA&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=1

2. In the second video,Loading a video source through webcam or some other camera is taught.

Link:-
https://www.youtube.com/watch?v=Jvf5y21ZqtQ&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=2

3.in this video, Drawing and writing on an existing image is taught.

Link:-
https://www.youtube.com/watch?v=U6uIrq2eh_o&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=3

4.in this video, basic image operations are taught such as region of image selection

Link:-
https://www.youtube.com/watch?v=1pzk_DIL_wo&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=4

5. in this video, image arithmetics and logic are taught.

Link:-
https://www.youtube.com/watch?v=_gfNpJmWIug&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=5

6.in this video,Thresholding is taught.

Link:-
https://www.youtube.com/watch?v=jXzkxsT9gxM&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=6

7.in this video,Color filtering is taught.

Link:-
https://www.youtube.com/watch?v=CCOXg75HkvM&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=7

8.in this video, Blurring and smoothing is taught.

Link:-
https://www.youtube.com/watch?v=sARklx6sgDk&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=8

9.in this video,Morphological transformations are taught.

Link:-
https://www.youtube.com/watch?v=YA5u2PI3hF0&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=9

10.in this video,Edge detection and Gradients are taught.

Link:-https://www.youtube.com/watch?v=CJMCoAsK-h0&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=10

11.in this video template matching is taught.

Link:-https://www.youtube.com/watch?v=2CZltXv-Gpk&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=11

12.in this video, Grabcut Foreground Extraction is taught.

Link:-https://www.youtube.com/watch?v=qxfP13BMhq0&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=12

13.in this video,Corner Detection is taught.

Link:-https://www.youtube.com/watch?v=6e6NbNegChU&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=13

14. in this video, Feature matching(Homography) by brute force method is taught.

Link:-https://www.youtube.com/watch?v=UquTAf_9dVA&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=14

15.in this video,MOG background reduction is taught.

Link:-https://www.youtube.com/watch?v=8-3vl71TjDs&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=15

16.in this video,Haar cascade Object detection for Face and Eye is taught.

Link:-https://www.youtube.com/watch?v=88HdqNDQsEk&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=16

17.in this video,an Introduction to make your own Haar cascade is taught.

Link:-https://www.youtube.com/watch?v=jG3bu0tjFbk&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=17

18.in this video,Gathering images for Haar cascade is taught.

Link:-https://www.youtube.com/watch?v=z_6fPS5tDNU&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=18

19.in this video,Cleaning images and creating description files is taught.

Link:-https://www.youtube.com/watch?v=t0HOVLK30xQ&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=19

20.in this video,training Haar cascade object detection is taught.

Link:-https://www.youtube.com/watch?v=eay7CgPlCyo&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=20

21.in this video,Haar cascade for image and video object classification is taught.

Link:-https://www.youtube.com/watch?v=-Mhy-5YNcG4&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq&index=21

Sentdex Tutorial series for Python programming.

This link contains the full tutorial for learning python programming language.

Link:-https://www.youtube.com/playlist?list=PLQVvvaa0QuDe8XSftW-RAxdo6OmaeL85M

INVENTORY:

SLOT NO.	COMPONENT	VOLTAGE RATING	QUANTITY
1	RASPBERRY PI ZERO	5V	1
2	16 GB SD CARD	-	1
3	SG90(SERVO)(TORQUE-2.5 kg cm)	4.8-6V	4
4	MG995(SERVO)(TORQUE- 11kg cm (6V))	4.8-7.2V	4
5	L293D	4.5-36V	2
6	L-BO MOTOR 150 RPM(TORQUE-2.5 kgf.cm)	3-9V	4
7	PI CAMERA ALONG WITH RIBBON	-	1
8	2200 MAH LI ON BATTERIES	4.2V	4
9	BUCK CONVERTOR	-	1
10	OMNI WHEELS 38mm	-	4
11	JUMPERS	-	-
12	MECHANICAL COMPONENTS : CHASSIS, LEGS, SERVO HOLDERS, SCREWS, BOLTS, SERVO HORNS	-	-

STEPS:

1.First we need to do a 'headless setup' of a raspberry pi.- https://hackernoon.com/raspberry-pi-headless-install-462ccabd75d0

2.Then we need to install Pigpio library on the raspberry pi.
http://abyz.me.uk/rpi/pigpio/download.html

3.The camera has to be activated in the raspberry pi configurations.

4.The dropbox uploader is setup in the raspberry pi. https://www.raspberrypi.org/magpi/dropbox-raspberry-pi/

5. The mechanical components have to be assembled.

6.Suitable connections with common ground,regulated voltage keeping in mind the voltage ratings of components used.

7.The suitable code needs to be composed (Appendix) and executed.

8.Sync a local directory with drop box.

9.Perform Image processing on laptop after installing suitable libraries.

APPENDIX:

Appendix A

CODE FOR LOCOMOTION:

```
import threading                        #for multi-threading,servo syncing
import os                               #for writing terminal commands
import time
os.system("sudo pigpiod")               #to launch GPIO(General Purpose Input Outpin) handling library
                                        ####AFTER PROGRAM IS OVER,WRITE "sudo killall pigpiod" TO STOP SENDING SIGNALS TO
COMPONENTS
import pigpio                           #library handling GPIO
pi= pigpio.pi()                         #creating object of library
#R-RIGHT    L-LEFT     U-UPPER     L-LOWER     F-FRONT     B-BACK
#FRONT IS THE SIDE WITH THE DOUBLE SIDE TAPE BETWEEN TWO SERVOS

RFU=2
RFL=3
RBU=7     #INVERT
RBL=17
LFU=27    #INVERT
LFL=22
LBU=10
LBL=9
#M-MOTOR    1=FORWARD    2=BACKWARD        3=PWM
MRF1=26
MRF2=21
MRF3=13

MRB1=5
MRB2=12
MRB3=19

MLF1=18
MLF2=15
MLF3=14

MLB1=24
MLB2=25
MLB3=23

#SETTING PINS TO OUTPUT MODE
```

```
pi.set_mode(RFU, pigpio.OUTPUT)
pi.set_mode(RFL, pigpio.OUTPUT)
pi.set_mode(RBU, pigpio.OUTPUT)
pi.set_mode(RBL, pigpio.OUTPUT)
pi.set_mode(LFU, pigpio.OUTPUT)
pi.set_mode(LFL, pigpio.OUTPUT)
pi.set_mode(LBU, pigpio.OUTPUT)
pi.set_mode(LBL, pigpio.OUTPUT)
pi.set_mode(MRF1, pigpio.OUTPUT)
pi.set_mode(MRF2, pigpio.OUTPUT)
pi.set_mode(MRF3, pigpio.OUTPUT)
pi.set_mode(MRB1, pigpio.OUTPUT)
pi.set_mode(MRB2, pigpio.OUTPUT)
pi.set_mode(MRB3, pigpio.OUTPUT)
pi.set_mode(MLB1, pigpio.OUTPUT)
pi.set_mode(MLB2, pigpio.OUTPUT)
pi.set_mode(MLB3, pigpio.OUTPUT)
pi.set_mode(MLF1, pigpio.OUTPUT)
pi.set_mode(MLF2, pigpio.OUTPUT)
pi.set_mode(MLF3, pigpio.OUTPUT)

#SETTING SEVOS TO BASE ANGLES
#FOR SERVOS 0-180 IS MAPPED ONTO 500-2500 WHILE CALLING FUNCTION

pi.set_servo_pulsewidth(RFU,660)
pi.set_servo_pulsewidth(RFL,2380)
pi.set_servo_pulsewidth(RBU,2500)
pi.set_servo_pulsewidth(RBL,1330)
pi.set_servo_pulsewidth(LFU,1440)
pi.set_servo_pulsewidth(LFL,1330)
pi.set_servo_pulsewidth(LBU,660)
pi.set_servo_pulsewidth(LBL,1330)

#SETTING ALL MOTORS TO STANDBY

pi.write(MRF1,0)
pi.write(MRF2,0)
pi.write(MRF3,0)
pi.write(MRB1,0)
```

```
pi.write(MRB2,0)

pi.write(MRB3,0)

pi.write(MLF1,0)

pi.write(MLF2,0)

pi.write(MLF3,0)

pi.write(MLB1,0)

pi.write(MLB2,0)

pi.write(MLB3,0)


def servo(pin,degree) :                                 #FUNCTION TAKES PIN NUMBER AND DEGREE(0-180) AS INPUT,CONVERTS TO 500-2500 AND ACTUATES SERVO

   degree=min(max(10*(int((degree*2000/180 + 500)/10)),500),2500)  #MAPPING WITH MAXIMUM AND MINIMUM CONSTRAINS ROUNDED OFF TO 10s

   while(pi.get_servo_pulsewidth(pin)>degree):

      pi.set_servo_pulsewidth(pin,pi.get_servo_pulsewidth(pin)-10)#SERVO MOVES IN SMALL STEPS AND TAKES SO THAT CHANGE IS NOT ABRUPT

      time.sleep(0.02)

   while(pi.get_servo_pulsewidth(pin)<degree):

      pi.set_servo_pulsewidth(pin,pi.get_servo_pulsewidth(pin)+10)

      time.sleep(0.02)


def height(h, tx, ty) :                                 #CENTRAL FUNCTION HANDLING ALL SERVOS,sevo() IS CALLED BY height() FOR ALL SERVOS POST CALCULATING ANGLES


    #THREADING HAS BEEN USED HERE,SO THAT ALL SERVOS ACTUATE AT THE SAME INSTANT RATHER THAN ONE AFTER ANOTHER,THE SMALL STEP CONCEPT MAKES THE DIFFERENCE NOTICIBLE

    t1=threading.Thread(target=servo,args=(RFU,min(max(15,(h+tx+ty)),75 )))

    t1.start()

    t2=threading.Thread(target=servo,args=(RFL,170-min(max(15,(h+tx+ty)),75)))

    t2.start()

    t3=threading.Thread(target=servo,args=(RBU,180-min(max(15,(h+tx-ty)),75)))

    t3.start()

    t4=threading.Thread(target=servo,args=(RBL,100-min(max(15,(h+tx-ty)),75)))

    t4.start()

    t5=threading.Thread(target=servo,args=(LFU,89-min(max(15,(h-tx+ty)),75)))

    t5.start()

    t6=threading.Thread(target=servo,args=(LFL,90-min(max(15,(h-tx+ty)),75)))

    t6.start()

    t7=threading.Thread(target=servo,args=(LBU,min(max(15,(h-tx-ty)),75)))

    t7.start()

    t8=threading.Thread(target=servo,args=(LBL,90-min(max(15,(h-tx-ty)),75)))

    t8.start()
```

```
def motor(pin1, pin2, pin3, s):     #FORWARD PIN1, BACKWARD PIN2 , PWM PIN3
                                #FUNCTION WHICH WRITES SPPED AND DIRECTIONT O A SPECIFIC MOTOR
    s=s*255/100                 #SPEED IS TAKEN IN RANGE 0-100 WHICH IS MAPPED TO 0-255


    if (s>0):
      pi.write(pin1, 1)
      pi.write(pin2, 0)
    elif(s<0):
      s = -s
      pi.write(pin2, 1)
      pi.write(pin1, 0)
    else:
      pi.write(pin1,0)
      pi.write(pin2,0)
    pi.set_PWM_dutycycle(pin3, s)


def holo(x,y,r):                            #POSITIVE R IS ANTI-CLOCKWISE
                                #FUNCTION WHICH CALCULATES SPEEDS OF ALL MOTORS AND CALLS motor() FOR EACH MOTOR
    motor(MRF1,MRF2,MRF3, (y-x)/(2**0.5)+r)
    motor(MLB1,MLB2,MLB3, (y-x)/(2**0.5)-r)
    motor(MLF1,MLF2,MLF3,1.15*((y+x)/(2**0.5)-r))  #1.15 AS THIS MOTOR HAD A HIGHER INTERNAL RESISITANCE
    motor(MRB1,MRB2,MRB3, (y+x)/(2**0.5)+r)


value=[0,0,0,0,0,0]
print("OPTIONS: \nX FOR X AXIS LOCOMOTION \nY FOR Y AXIS LOCOMOTION \nR FOR ANTI CLOCKWISE ROTATION \nH
FOR HEIGHT ADJUST \nTX FOR X TILT \nTY FOR Y TILT \nFORMAT : <PARAMETER> <VALUE> <PARAMETER> <VALUE>")
while 1:
   para = input('Enter parameter\n').split(' ')
   parameters = ['X','Y','R','H','TX','TY']
   for item in para:
            if item in parameters:
               try:
                  value[parameters.index(item)]=int(para[para.index(item)+1])
               except:
                  print("WRONG INPUT FORMAT")
   print(*value)
   height(value[3],value[4],value[5])
      holo(value[0],value[1],value[2])
```

Appendix B:

CODE FOR LIVE STREAMING FOOTAGE:

```
# Web streaming example
# Source code from the official PiCamera package
# http://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming

import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Surveillance Camera</h1></center>
<center><img src="stream.mjpg" width="640" height="480"></center>
</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
```

```
        return self.buffer.write(buf)


class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()
```

```
class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()
    #Uncomment the next line to change your Pi's Camera rotation (in degrees)
    camera.rotation = 180
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()
```

Appendix C:
CODE FOR EYE, FACE DETECTION:

```
import cv2
#imports the cv2 library and enables it to be used in this python program.
import numpy as np
#imports the numpy library and enables it to be used in this python program.
import datetime
#imports the datetime library and enables it to be used in this python program.
import time
#imports the timelibrary and enables it to be used in this python program.

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
#this destination contains the cascade for face detection.
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade_eye.xml
#this destination contains the cascade for eye detection.

fourcc = cv2.VideoWriter_fourcc(*'XVID')
```

```
# XVID is a codec for the video file. this lines enables saving of video files.
out1 = cv2.VideoWriter('VIDEO FEED OUTPUT\Output%3d.avi' % 1,fourcc,10.0,(640,480))
out2 = cv2.VideoWriter('VIDEO FEED OUTPUT\Output%3d.avi' % 2,fourcc,10.0,(640,480))
# creates a video file storing the processed video feed enabled with face, eyes and motion detection.


for n in range (1000):
   #loops from 0 to 999.
   try:
       cap = cv2.VideoCapture('VIDEO FEED\input%3d.avi' % (n))
      # captures the video feed from the folder VIDEO FEED.
      fgbg = cv2.createBackgroundSubtractorMOG2()
      itime=time.time()
      # stores the value of time before the execution of the while loop.
      while ((time.time()-itime) < 10) :
         ret, img = cap.read()
         #captures the video feed and stores it in img variable.
         fgmask = fgbg.apply(img)
         # this is for enabling motion detection.
          gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         #this is for grayscaling the video feed.
         faces = face_cascade.detectMultiScale(gray, 1.3, 5)
         #for face detection.


         x,y,w,h = 0,0,0,0
         for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            roi_gray = gray[y:y+h+25, x:x+w+25]
            roi_color = img[y:y+h+25, x:x+w+25]
            # creates a rectangle around the face.


            eyes = eye_cascade.detectMultiScale(roi_gray)
            #detects eyes.
            if len(eyes) <2*len(faces)+1 :
               for (ex,ey,ew,eh) in eyes:
                  cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
            # prevents the presence of more than 2 eyes in 1 face and also creates a rectangle around the eye


        img2 = cv2.medianBlur(cv2.cvtColor(fgmask,cv2.COLOR_GRAY2RGB),9)#9 IS SENSITIVITY COUNTER,DECREASE TO
MAKE SENSITIVE,USE ODD NUMBERS ONLY
         #this is executed to grayscale the video feed.
         currentDT = datetime.datetime.now()
```

```
        font = cv2.FONT_HERSHEY_SIMPLEX
        #font in which date and time will be displayed.

        cv2.putText(img2,currentDT.strftime("%Y-%m-%d %H:%M:%S"),(450,420), font, 0.5,(255,255,255),2,cv2.LINE_AA)
        cv2.putText(img,currentDT.strftime("%Y-%m-%d %H:%M:%S"),(450,420), font, 0.5,(255,255,255),2,cv2.LINE_AA)
        #prints the current date and time on the live video feed and on the processed video feed.

        if img2.any()!= 0:
            out2.write(img2)

        ctime=time.time()
        if len(faces) != 0:
            out1.write(img)
            itime=time.time()
        elif ctime-itime<10:
            out1.write(img)
        else  :
            out1.release()
            out2.release()

            cap.release()
            cv2.destroyAllWindows()
            break;

        cv2.imshow('videofeed_fgmask',img)
        # Displays the video feed in which eye and face detection is being performed.
        cv2.imshow('fgmask',img2)
        # Displays the videofeed in which motion detection is being performed.
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break;
    except:
        continue
        #exits the loop if this set is executed. it signifies that the video files are no longer available.
out1.release()
#stops writing the video file of face and eye detection.
out2.release()
#stops writing the video file of motion detection.
cap.release()
# stops capturing the live/downloaded video feed.
```

```
cv2.destroyAllWindows()
#destroys all windows.
```

Appendix D:
CODE FOR DROP BOX UPLOADER:

```
import os
import subprocess
from subprocess import Popen, PIPE
#The directory to sync
syncdir="/home/pi/cctv/"
#Path to the Dropbox-uploaded shell script
uploader = "/home/pi/Dropbox-Uploader/dropbox_uploader.sh"
#If 1 then files will be uploaded. Set to 0 for testing
upload = 1
#If 1 then don't check to see if the file already exists just upload it, if 0 don't upload if already exists
overwrite = 0
#If 1 then crawl sub directories for files to upload
recursive = 1
#Delete local file on successfull upload
deleteLocal = 0
#Prints indented output
def print_output(msg, level):
    print((" " * level * 2) + msg)

#Gets a list of files in a dropbox directory
def list_files(path):
    p = Popen([uploader, "list", path], stdin=PIPE, stdout=PIPE, stderr=PIPE)
    output = p.communicate()[0].decode("utf-8")
    fileList = list()
     lines = output.splitlines()
    for line in lines:
        if line.startswith(" [F]"):
            line = line[5:]
            line = line[line.index(' ')+1:]
            fileList.append(line)
    return fileList
```

```
#Uploads a single file
def upload_file(localPath, remotePath):
    p = Popen([uploader, "upload", localPath, remotePath], stdin=PIPE, stdout=PIPE, stderr=PIPE)
    output = p.communicate()[0].decode("utf-8").strip()
    if output.startswith("> Uploading") and output.endswith("DONE"):
        return 1
    else:
        return 0


#Uploads files in a directory
def upload_files(path, level):
    fullpath = os.path.join(syncdir,path)
    print_output("Syncing " + fullpath,level)
    if not os.path.exists(fullpath):
        print_output("Path not found: " + path, level)
    else:
        #Get a list of file/dir in the path
        filesAndDirs = os.listdir(fullpath)
        #Group files and directories
        files = list()
        dirs = list()
        for file in filesAndDirs:
            filepath = os.path.join(fullpath,file)
            if os.path.isfile(filepath):
                files.append(file)
            if os.path.isdir(filepath):
                dirs.append(file)
        print_output(str(len(files)) + " Files, " + str(len(dirs)) + " Directories",level)
        #If the path contains files and we don't want to override get a list of files in dropbox
        if len(files) > 0 and overwrite == 0:
            dfiles = list_files(path)
        #Loop through the files to check to upload
        for f in files:
            print_output("Found File: " + f,level)
            if upload == 1 and (overwrite == 1 or not f in dfiles):
                fullFilePath = os.path.join(fullpath,f)
                relativeFilePath = os.path.join(path,f)
                print_output("Uploading File: " + f,level+1)
                if upload_file(fullFilePath, relativeFilePath) == 1:
```

```
                print_output("Uploaded File: " + f,level + 1)
                if deleteLocal == 1:
                    print_output("Deleting File: " + f,level + 1)
                    os.remove(fullFilePath)
            else:
                print_output("Error Uploading File: " + f,level + 1)
    #If recursive loop through the directories
    if recursive == 1:
        for d in dirs:
            print_output("Found Directory: " + d, level)
            relativePath = os.path.join(path,d)

            upload_files(relativePath, level + 1)
#Start
upload_files("",1)
```

Appendix D:

Sketch