**DESIGN AND DEVELOPMENT OF LOW COST THERMAL IMAGING SYSTEM (OPERATED BY GUI) ALONG WITH DIAGNOSIS OF DRY EYE (KERATOCONJUCTIVITIS SICCA) IN HUMANS**

**A PROJECT REPORT**

*Submitted by*

**MANGALESWARA SIDDARTH M 2016116022**

**SANJEEVI G 2016116040**

**YOGESH KUMAR K R 2016116055**

**B.E BIOMEDICAL ENGINEERING, FINAL YEAR.**

**PROBLEM STATEMENT**

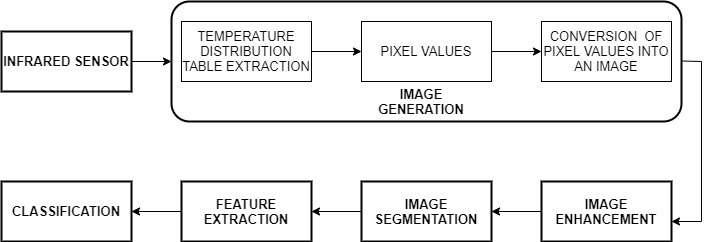
**NEED FOR THE PROJECT**

The available devices for medical applications are highly expensive. The thermal imaging system can be used in addition to any other medical imaging device for better diagnosing purposes. The physiological variations in human body with respect to temperature can be diagnosed easily with thermal imaging system. But, it is not possible with the novel medical imaging devices like X-ray, CT and MRI since it reveals only the physical changes.

The physiological changes can only be monitored using Infrared red imaging system**.** The traditional methods like Schirmer’s test is invasive, hence a new non-invasive method is in need, and infrared thermography is an effective non-invasive way for improved diagnosis.

## OBJECTIVE OF THE PROJECT

The main objective of this project is to design and develop a low cost thermal imaging system for medical applications and compare it with the standardized infra-red thermal imaging system to check the system’s efficiency in terms of accuracy percentages. Another objective is to capture eye images, perform image enhancement, segmentation, and extract various features like are Optical Surface temperature of entire eye, Optical Surface Temperature of Cornea and Temperature Deviation along Cornea, and classify them as normal or dry eye images based on threshold found in research papers. The accuracy, sensitivity, specificity calculation of these features are also found as part of our work.

**BLOCK DIAGRAM:**

**IMPLEMENTATION DETAILS:**

The methodology consists of four blocks, which are handpicked from research papers. The image generation block consists of extraction of temperature distribution value extraction, pixel value conversion into an image. The image enhancement includes image sharpening, K means cluster grouping, whereas VIBGYOR segmentation is used. The extracted features are OST (Optical Surface Temperature) of entire eye, OST of Cornea, and TDC (Temperature Deviation along Cornea). The components used are:

## ADAFRUIT AMG8833 IR THERMAL CAMERA

1. **RASPBERRY PI**
2. **RASPBERRY PI CAMERA**
3. **SMART PHONE WITH MOBILE VNC APPLICATION INSTALLED**

## PRINCIPLE:

The Camera passively measures heat radiation from an infrared-emitting grey body. The temperature is calculated by employing the Stefan-Boltzmann law:

***Π = εAσT 4***

*ε* is called the emissivity (between 0 and 1), *A* is the surface area, *σ* is the Stefan-Boltzmann constant, *T* is the temperature of the body, and *Π* is the radiative power.

The thermistor measures the temperature change and converts it into analog voltage. The analog voltage is converted into temperature using the below formula

***V ≈ k (Tobj4 - Ts4)***

*V* above represents the voltage measured by the raw sensor. The variable *k* is an empirical constant that absorbs the *A, ε, σ* and electronic noise that may exist. The *Ts* is the temperature of the sensor itself, and the remaining *Tobj* is the temperature of the object being measured.

The Analog to digital Converter converts the analog temperature into digital values and sends it to the microcontroller via I2C() Using SDA(), and SCL() pins.

***Tobj ≈ (V/k + Ts4) ¼***

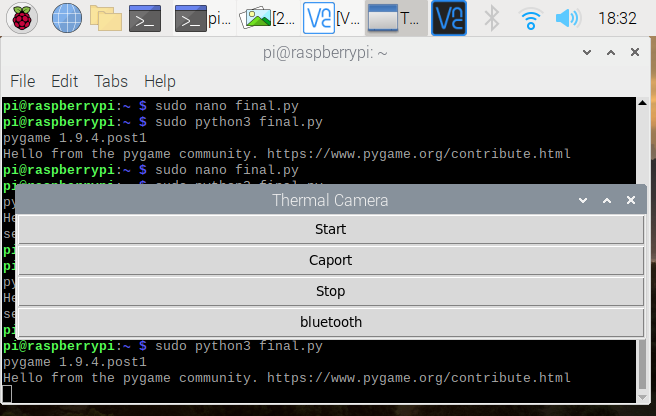
This is the final approximation equation, an analog to digital converter converts the analog temperature values to digital and sends it microprocessor, which receives it through I2C.

## GUI CREATION:

An application is created using Tkinter library. In this we have created 3 buttons namely,

* Start
* Caport (capture+export)
* Stop

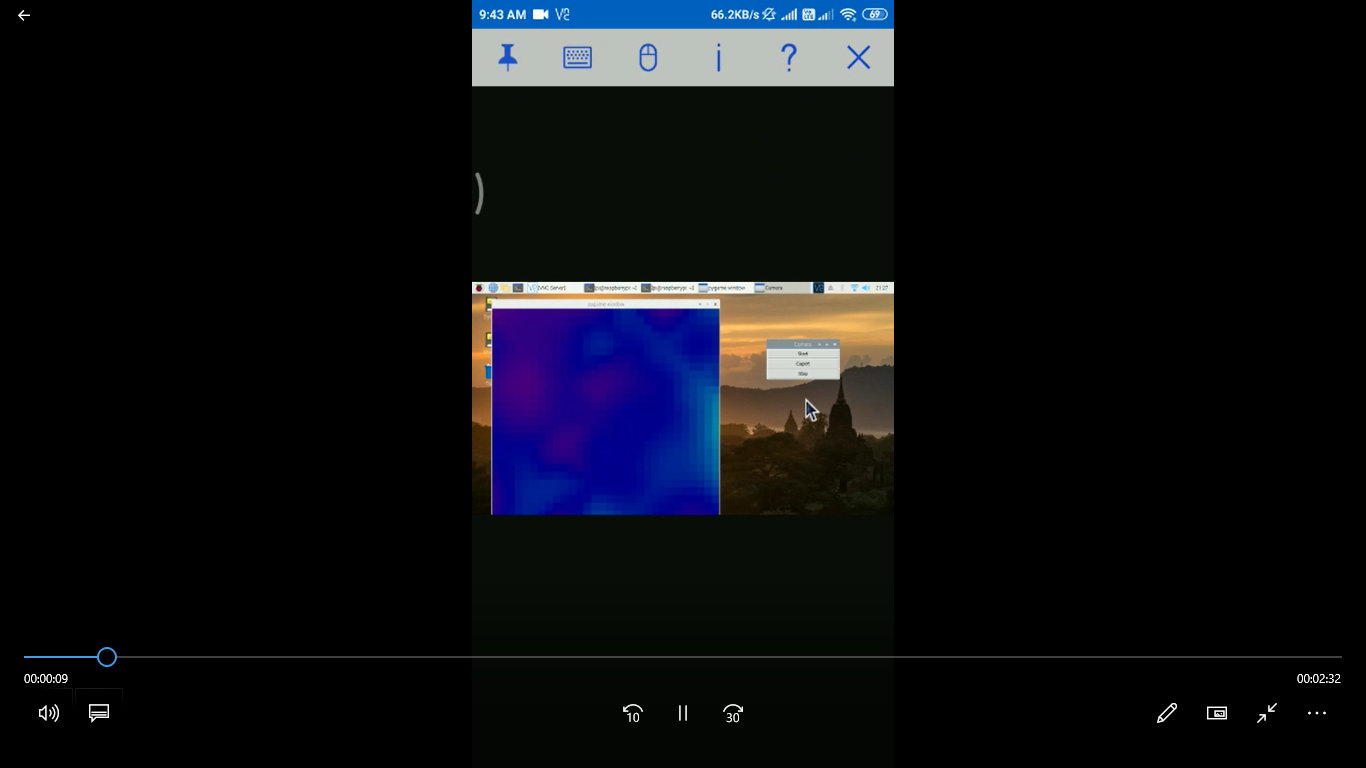
On pressing start button, the camera starts showing the thermal image. On pressing Caport button, the camera captures the thermal image, as well as normal image using pi camera and stores the temperature distribution value as CSV file (Comma Separated Value). The stop button quits the interface.



**GUI CREATION**

## MOBILE VNC FOR ANDROID:

In our project, we use mobile VNC to remotely control and transfer data from microprocessor to smartphone. This enables us to view as well as control the thermal imaging system directly using our smartphone via the GUI built.

****

**MOBILE VNC**

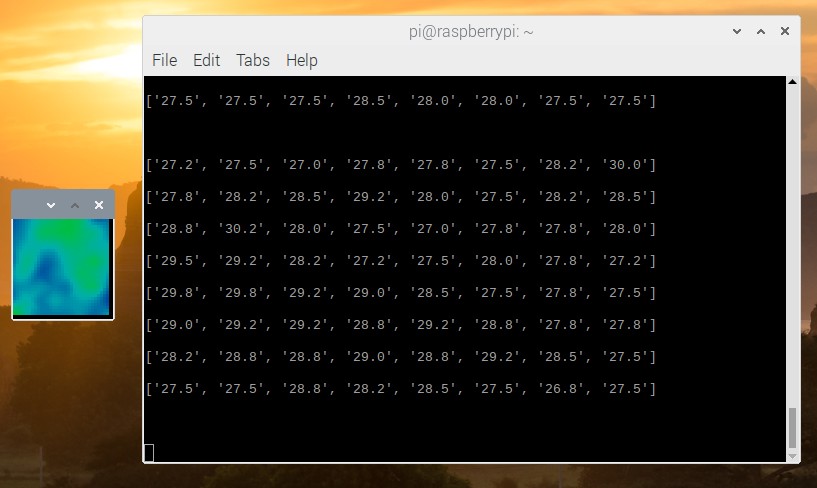
CAPTURING EYE IMAGES:

The images are taken in 250C at a distance of 3cm from the imaging system. Temperature ranges is selected as 31.2 to 33.C, taken as per research paper. The eye images are taken in such a way that they are placed in the center of the entire camera. Right and left images are taken along with normal images using raspberry pi camera with 5 Mega Pixel clarity. They are further processed to extract features and classify them accordingly. Firstly, the thermistor sensor sends the 64 temperature pixel values, which is converted into a pseudo-color image of 240\*240 pixels by means of bipolar interpolation with a capturing range of 16sqcm.

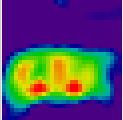
**RESULTS**

**IMAGE GENERATION**

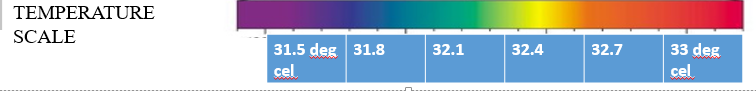
For dry eye capturing purpose, temperature distribution 31.2-330 C is chosen scientifically based on a research paper. Firstly, the thermistor sensor sends the 64 temperature pixel values, which is converted into a pseudo-color image of 240\*240 pixels by means of bipolar interpolation with a capturing range of 16sqcm.

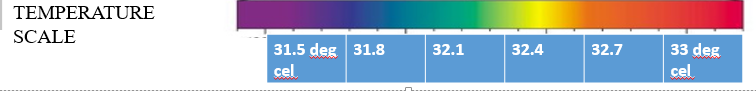


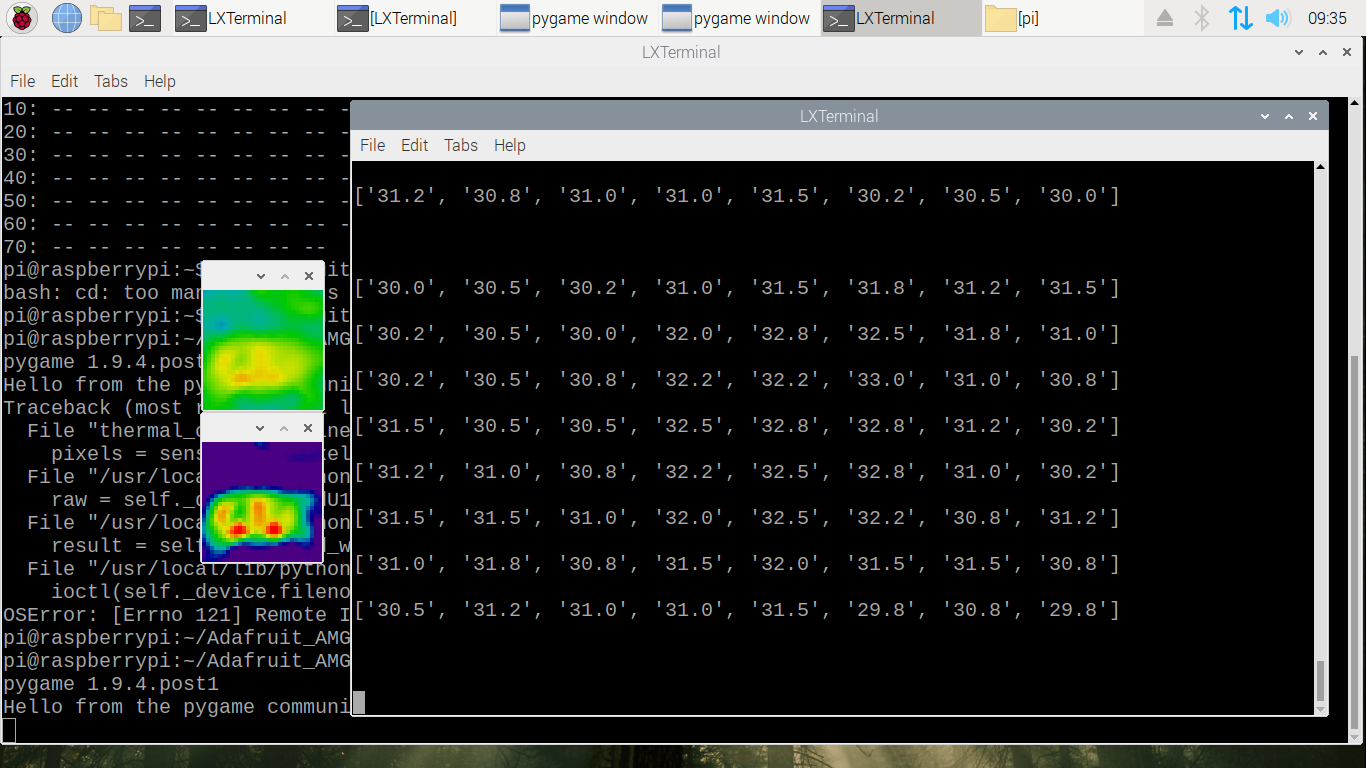
## THERMAL IMAGE OF HAND WITH PIXEL VALUES

**SUBJECT 1:**

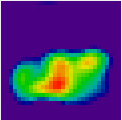
**RIGHT EYE THERMAL IMAGE RIGHT EYE NORMAL IMAGE**

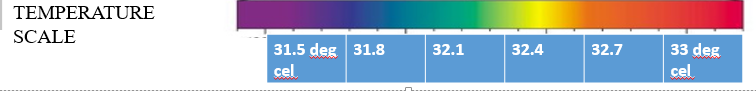


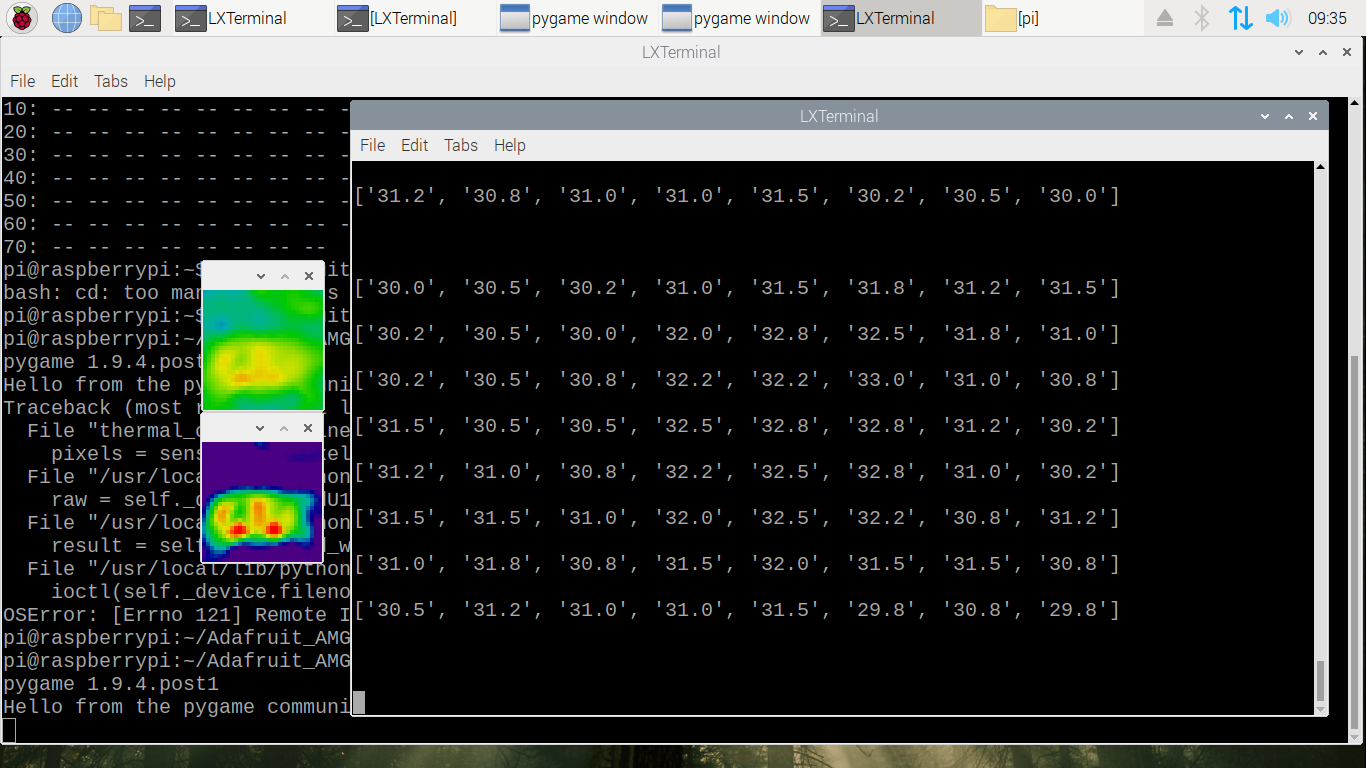




**TEMPERATURE MATRIX OF GIVEN THERMAL IMAGE**

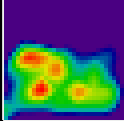
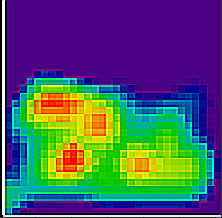
 **LEFT EYE THERMAL IMAGE LEFT EYE NORMAL IMAGE**



**TEMPERATURE MATRIX OF GIVEN THERMAL IMAGE**

**IMAGE ENHANCEMENT**

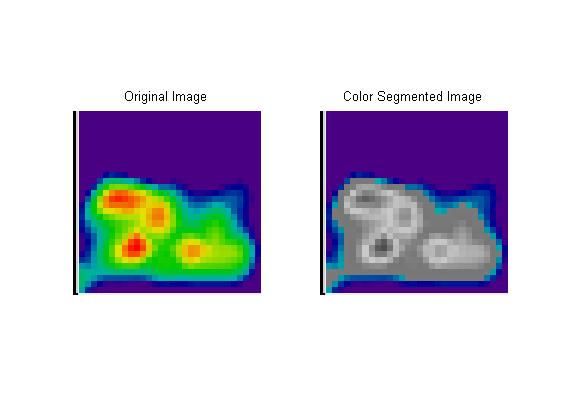
**IMAGE SHARPENING**

****

Original image After image Sharpening

**IMAGE SEGMENTATION**

**K Means Clustering Segmentation (Color Grouping)**

****

**K means segmented image**

**FEATURE EXTRACTION AND CLASSIFICATION**

Features are extracted as per the formula given in base papers we have studied. The main features extracted are:

* OST (Optical Surface Temperature) of entire eye
* OST of Cornea
* TDC (Temperature Deviation along Cornea)

## 1. OPTICAL SURFACE TEMPERATURE OF ENTIRE EYE

The entire eye OST is calculated after separation of foreground eye image from background. The number of color pixels are multiplied with individual temperature values. These values after division by total number of pixels yield average Optical surface temperature of entire eye.

Formula:

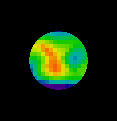
* Sum=((33\*R)+(32.7\*O)+(32.4\*Y)+(32.1\*G)+(31.8\*B))
* Divisor=R+O+Y+G+B
* OST=Sum/Divisor=temperature
* R,G,B,Y,O- no. of respective colour pixels

## CLASSIFICATION OF EYE BASED ON OST OF EYE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subject no: | Left eye temperature (OST) in degree Celsius: | Right eye temperature (OST) in degree Celsius: | Difference in degree Celsius: | Classification |
| 1 | 32.51 | 32.52 | 0.01 | Normal |
| 2 | 32.61 | 32.64 | 0.02 | Normal |
| 3 | 32.55 | 32.57 | 0.04 | Normal |
| 4 | 32.81 | 32.63 | 0.18 | Dry eye |
| 5 | 32.72 | 32.64 | 0.08 | Normal |
| 6 | 32.73 | 32.71 | 0.02 | Normal |
| 7 | 32.91 | 32.87 | 0.04 | Normal |
| 8 | 32.73 | 32.56 | 0.17 | Dry eye |
| 9 | 32.61 | 32.51 | 0.1 | Normal |
| 10 | 32.44 | 32.66 | 0.22 | Dry eye |

For classification, the threshold for difference has been set as **0.15oC**. This is primarily done based on grouping, and it is done as an analogy using the research paper.

## 2. OPTICAL SURFACE TEMPERATURE OF CORNEA

According to a research paper, radius of cornea is approximately equal to 1/4th or 25% of entire eye. A Temperature difference of **0.6oC** or above between right and left cornea indicates presence of dry eye. This asymmetrical property of cornea is found in patients with dry eye. A mask is created for all images with radius 25% of entire eye, and temperature of cornea part alone is calculated using computer vision library. The same formula is used here to find temperature as in case entire eye OST feature. The mask is created using Computer vision library by taking radius as 25% of entire eye length. All other areas of the image other than cornea is masked.

**MASKED IMAGE:**

CLASSIFICATION OF EYE BASED ON OST OF CORNEA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subject no: | Left eye Cornea temperature(OST) in degree Celsius: | Right eye Cornea temperature(OST) in degree Celsius: | Difference in degree Celsius: | Classification |
| 1 | 32.96 | 32.95 | 0.01 | Normal |
| 2 | 32.99 | 32.94 | 0.05 | Normal |
| 3 | 32.97 | 32.98 | 0.01 | Normal |
| 4 | 32.87 | 32.23 | 0.64 | Dry eye |
| 5 | 32.97 | 32.98 | 0.01 | Normal |
| 6 | 32.96 | 32.93 | 0.03 | Normal |
| 7 | 32.91 | 32.89 | 0.02 | Normal |
| 8 | 32.91 | 32.31 | 0.6 | Dry eye |
| 9 | 32.95 | 32.97 | 0.02 | Normal |
| 10 | 32.92 | 32.91 | 0.01 | Normal |

## 3. TEMPERATURE DEVIATION ALONG CORNEA

This feature is extracted by creating a line along the diameter of the Cornea, and measuring the temperature change. The research paper shows an accuracy of 0.010C, but our camera’s accuracy is 0.250C. They had stated that 17 temperature variations as normal, which translates to 5.10C. All values below this threshold are considered to be normal and those above this are considered as abnormal.

## CLASSIFICATION OF EYE BASED ON TEMPERATURE DEVIATION ALONG CORNEA

|  |  |  |
| --- | --- | --- |
| Subject no: | Temperature Deviation along Cornea: | Classification |
| 1 | 4.4 | Normal |
| 2 | 4.5 | Normal |
| 3 | 4.5 | Normal |
| 4 | 5.3 | Dry eye |
| 5 | 4.5 | Normal |
| 6 | 4.4 | Normal |
| 7 | 4.4 | Normal |
| 8 | 5.4 | Dry eye |
| 9 | 4.4 | Normal |
| 10 | 4.5 | Normal |



**MASKED IMAGE WITH A LINE DRAWN ALONG THE CORNEA**

Complete thermal imaging system is built with application and output is displayed on smart phone. Image enhancement, segmentation, and feature extraction is complete. Our window has 900 pixels in display. Images are taken for **10 test subjects** with normal, and complete findings are listed in a table. Dry eye images are taken by simulating environmental conditions. Features like Temperature Deviation along Cornea and Cornea Optical Surface Temperature are also completed. They are verified by using VIBGYOR based segmentation.

**VIBGYOR SEGMENTATION:**

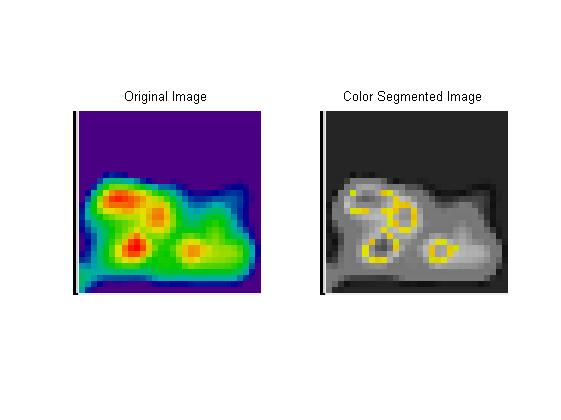
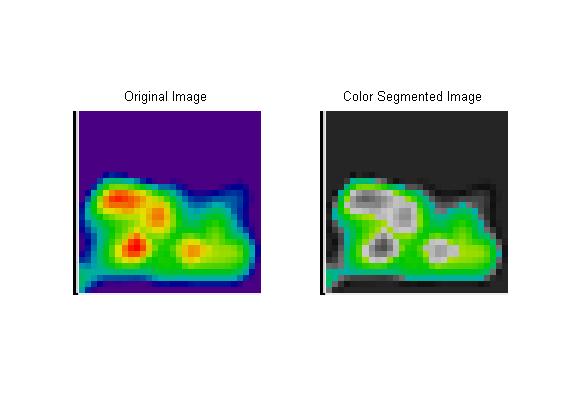
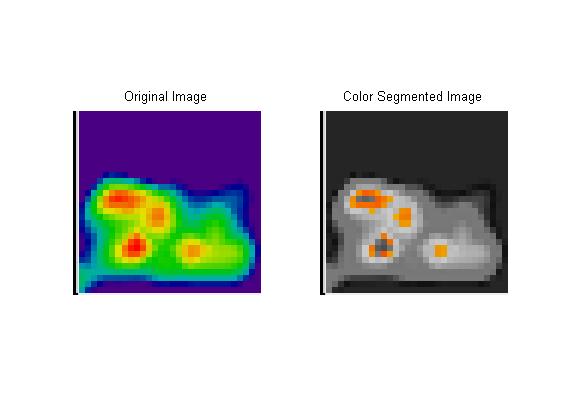
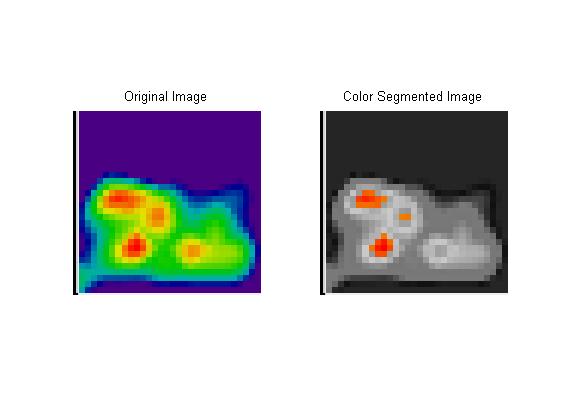
VIBGYOR based segmentation is done to separate individual colours.. The VIBGYOR based segmentation can also be used to detect a wide variety of pixels other than mentioned colours. Firstly, the images are converted to RGB from HSV format and based on certain filter function values and choices given to separate the individual colours, the output yields an image where the corresponding colours are highlighted and all the rest of the colours turned into grey. This is used, since it is the most effective way to extract features like OST (Optical Surface Temperature) of entire eye, OST of Cornea, TDC (Temperature Deviation along Cornea). This used for verification purposes.

## 

**VIBGYOR IMAGE VIOLET SEGMENTED IMAGE**

**INDIGO SEGMENTED IMAGE BLUE SEGMENTED IMAGE**

**GREEN SEGMENTED IMAGE YELLOW SEGMENTED IMAGE**

 **GREEN SEGMENTED IMAGE YELLOW SEGMENTED IMAGE** 

**ORANGE SEGMENTED IMAGE RED SEGMENTED IMAGE**

**CONCLUSION**

A low cost thermal imaging system is built using AMG88xx, Raspberry Pi-3b and output is displayed in smart phone. The entire camera can be controlled with the help of smart phone. The normal image, Thermal image and temperature values in CSV file format can be exported to smartphone via Bluetooth. The total cost of the thermal imaging system is very cheap. The results are as much as accurate compared to the standard thermal imaging devices in the market. The obtained dry eye thermal images with the temperature distribution provides us a better diagnosis of the condition. Since, it is non-invasive this way of diagnosis is more preferred on comparison to other methods.

Python assignment code:

"""

GUI CREATED WITH START, CAPORT, STOP

Caport captures the image and for temperature csv values suitable for export along with capture of normal image

"""

import tkinter as tk

import time

import board

import busio

import board

import adafruit\_amg88xx

from Adafruit\_AMG88xx import Adafruit\_AMG88xx

import pandas as pd

import pygame

import os

import math

import bluetooth

import numpy as np

from scipy.interpolate import griddata

from colour import Color

import picamera

win= tk.Tk()

win.title("Camera")

on=True

def sensor():

global on

on=True

#low range of the sensor (this will be blue on the screen)

MINTEMP = 28

#high range of the sensor (this will be red on the screen)

MAXTEMP = 34

#how many color values we can have

COLORDEPTH = 1024

os.putenv('SDL\_FBDEV', '/dev/fb1')

pygame.init()

#initialize the sensor

sensor = Adafruit\_AMG88xx()

points = [(math.floor(ix / 8), (ix % 8)) for ix in range(0, 64)]

grid\_x, grid\_y = np.mgrid[0:7:32j, 0:7:32j]

#sensor is an 8x8 grid so lets do a square

height = 720

width = 720

#the list of colors we can choose from

blue = Color("indigo")

colors = list(blue.range\_to(Color("red"), COLORDEPTH))

#create the array of colors

colors = [(int(c.red \* 255), int(c.green \* 255), int(c.blue \* 255)) for c in colors]

displayPixelWidth = width / 30

displayPixelHeight = height / 30

lcd = pygame.display.set\_mode((width, height))

lcd.fill((255,0,0))

pygame.display.update()

pygame.mouse.set\_visible(False)

lcd.fill((0,0,0))

pygame.display.update()

#some utility functions

def constrain(val, min\_val, max\_val):

return min(max\_val, max(min\_val, val))

def map(x, in\_min, in\_max, out\_min, out\_max):

return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min

#let the sensor initialize

time.sleep(.1)

x=100

while (x):

#read the pixels

pixels = sensor.readPixels()

pixels = [map(p, MINTEMP, MAXTEMP, 0, COLORDEPTH - 1) for p in pixels]

#perdorm interpolation

bicubic = griddata(points, pixels, (grid\_x, grid\_y), method='cubic')

#draw everything

for ix, row in enumerate(bicubic):

for jx, pixel in enumerate(row):

pygame.draw.rect(lcd, colors[constrain(int(pixel), 0, COLORDEPTH- 1)], (displayPixelHeight \* ix, displayPixelWidth \* jx, displayPixelHeight, displayPixelWidth))

pygame.display.update()

x=x-1

#win.update(10)

def export():

global on

on=True

i2c = busio.I2C(board.SCL, board.SDA)

amg = adafruit\_amg88xx.AMG88XX(i2c)

time.sleep(1)

# Create an empty list and append row temperature values

lst=[]

for row in amg.pixels:

lst.append(row)

df = pd.DataFrame(lst)

print (df)

df.to\_csv('sid.csv')

# normal image capturing

camera=picamera.PiCamera()

camera.capture('normal.jpg')

#IR image capture

MINTEMP = 28

MAXTEMP = 34

COLORDEPTH = 1024

os.putenv('SDL\_FBDEV', '/dev/fb1')

pygame.init()

sensor = Adafruit\_AMG88xx()

points = [(math.floor(ix / 8), (ix % 8)) for ix in range(0, 64)]

grid\_x, grid\_y = np.mgrid[0:7:32j, 0:7:32j]

height = 720

width = 720

blue = Color("indigo")

colors = list(blue.range\_to(Color("red"), COLORDEPTH))

colors = [(int(c.red \* 255), int(c.green \* 255), int(c.blue \* 255)) for c in colors]

displayPixelWidth = width / 30

displayPixelHeight = height / 30

lcd = pygame.display.set\_mode((width, height))

lcd.fill((255,0,0))

pygame.display.update()

pygame.mouse.set\_visible(False)

lcd.fill((0,0,0))

pygame.display.update()

def constrain(val, min\_val, max\_val):

return min(max\_val, max(min\_val, val))

def map(x, in\_min, in\_max, out\_min, out\_max):

return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min

time.sleep(.1)

if on:

pixels = sensor.readPixels()

pixels = [map(p, MINTEMP, MAXTEMP, 0, COLORDEPTH - 1) for p in pixels]

bicubic = griddata(points, pixels, (grid\_x, grid\_y), method='cubic')

for ix, row in enumerate(bicubic):

for jx, pixel in enumerate(row):

pygame.draw.rect(lcd, colors[constrain(int(pixel), 0, COLORDEPTH- 1)], (displayPixelHeight \* ix, displayPixelWidth \* jx, displayPixelHeight, displayPixelWidth))

pygame.display.update()

pygame.image.save(lcd, "thermal image.jpeg")

def off():

global on

on=False

print("sensor is switched off")

win.quit()

button = tk.Button(win, text='Start', width=25, command=sensor)

button.pack()

button = tk.Button(win, text='Caport', width=25, command=export)

button.pack()

button = tk.Button(win, text='Stop', width=25, command=off)

button.pack()

tk.mainloop()

"""

Image enhancement by sharpening

"""

from PIL import Image

from PIL import ImageFilter

import cv2

# Open an already existing image

imageObject = Image.open("thermal image.jpeg");

imageObject.show();

# Apply sharp filter

sharpened1 = imageObject.filter(ImageFilter.SHARPEN);

sharpened2 = sharpened1.filter(ImageFilter.SHARPEN);

# Show the sharpened images

sharpened1.show();

sharpened2.show();

cv2.imwrite('sharpened img.png',sharpened2)

"""

K means clustering and Optical surface temperature of entire eye feature

"""

from PIL import Image

import colorsys

import matplotlib.pyplot as plt

import numpy

import cv2

original\_image = cv2.imread('sharpened img.png')# read an image

img=cv2.cvtColor(original\_image,cv2.COLOR\_BGR2RGB)# convert it from bgr to rgb

vectorized = img.reshape((-1,3))

vectorized = np.float32(vectorized)

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)

K = 7

attempts=10

ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS\_PP\_CENTERS)

center = np.uint8(center)

res = center[label.flatten()]

result\_image = res.reshape((img.shape))

figure\_size = 15

result\_image=cv2.cvtColor(result\_image,cv2.COLOR\_BGR2RGB)

cv2.imwrite('segment ir.png',result\_image)

plt.figure(figsize=(figure\_size,figure\_size))

plt.subplot(1,2,1),plt.imshow(img)

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(1,2,2),plt.imshow(result\_image)

plt.title('Segmented Image when K = %i' % K), plt.xticks([]), plt.yticks([])

plt.show()

im = Image.open('segment ir.png')

NUM\_BUCKETS = 6 # 6 main colors are used

colour\_counts = [0] \* NUM\_BUCKETS

for pixel in im.getdata():

hue, saturation, value = colorsys.hsv\_to\_rgb(pixel[0], pixel[1], pixel[2])

hue\_bucket = hue \* NUM\_BUCKETS // 255 # Using python3 to get an int

colour\_counts[hue\_bucket] += 1

colour\_names = ["red", "yellow", "green", "cyan", "blue", "magenta"]

for name, count in [x for x in zip(colour\_names, colour\_counts)]:

print("{n} = {c}".format(n=name, c=count))

plt.imshow(im)

# separate foreground from background

backg = 0

real = 0

for pixel in im.getdata():

if pixel == (0,0,0 ): # if your image is RGB (if RGBA, (0, 0, 0, 255) or so

backg += 1

else:

real += 1

print('backg=' + str(backg)+', fore='+str(real))

plt.imshow(im)

# for corresponding colors given above in gui, backtracing temperature values from color image

b=int(31.8\*colour\_counts[4])

g=int(32.1\*colour\_counts[2])

y=int(32.4\*colour\_counts[1])

r=int(33\*colour\_counts[0])

sum=b+g+y+r

print("sum=",sum)

div= int(colour\_counts[4]+colour\_counts[2]+colour\_counts[1]+colour\_counts[0])

print("divident=",div)

avg\_temp=sum/div

print("avg temperature=",avg\_temp)

plt.imshow(im)

"""

Feature2- Application of cornea mask and Cornea Optical Surface Temperature

"""

from PIL import Image

import colorsys

import matplotlib.pyplot as plt

import cv2

import numpy as np

im=cv2.imread('segment ir.png') #read an image

# A mask is created for all images with radius 25% of entire eye, and temperature of cornea part alone is calculated

# creation of mask

h, w = im.shape[:2]

Y, X = np.ogrid[:h, :w]

center = (int(w/2), int(h/2))

H=int(h/2)

radius=w/4

dist\_from\_center = np.sqrt((X - center[0])\*\*2 + (Y-center[1])\*\*2)

mask = dist\_from\_center >= radius

masked\_im = im.copy()

masked\_im[mask]=0

#plt.imshow(masked\_im)

a=cv2.imwrite('cornea\_sept.png',masked\_im)

im = Image.open('cornea\_sept.png')

NUM\_BUCKETS = 6 # 6 basic colors used

colour\_counts = [0] \* NUM\_BUCKETS

for pixel in im.getdata():

hue, saturation, value = colorsys.hsv\_to\_rgb(pixel[0], pixel[1], pixel[2])

hue\_bucket = hue \* NUM\_BUCKETS // 255 # Using python3 to get an int

colour\_counts[hue\_bucket] += 1

colour\_names = ["red", "yellow", "green", "cyan", "blue", "magenta"]

for name, count in [x for x in zip(colour\_names, colour\_counts)]:

print("{n} = {c}".format(n=name, c=count))

masked\_im=cv2.cvtColor(masked\_im, cv2.COLOR\_BGR2RGB)

plt.imshow(masked\_im)

b1=int(31.8\*colour\_counts[4])

g1=int(32.1\*colour\_counts[2])

y1=int(32.4\*colour\_counts[1])

r1=int(33\*colour\_counts[0])

sum1=b1+g1+y1+r1

print("sum=",sum)

div1= int(colour\_counts[4]+colour\_counts[2]+colour\_counts[1]+colour\_counts[0])

print("divident=",div)

avg\_temp1=sum1/div1

print("cornea temperature=",avg\_temp1)

"""

Feature 3- Temeprature Deviation along Cornea

"""

from PIL import Image

import colorsys

import matplotlib.pyplot as plt

import cv2

import numpy as np

im = cv2.imread('segment ir.png')

h, w = im.shape[:2]

Y, X = np.ogrid[:h, :w]

print(h,w)

center = (int(w/2), int(h/2))

H=int(h/2)

radius=w/4

dist\_from\_center = np.sqrt((X - center[0])\*\*2 + (Y-center[1])\*\*2)

#print(dist\_from\_center)

mask = dist\_from\_center >= radius

masked\_im = im.copy()

l=masked\_im[mask]=0

plt.imshow(masked\_im)

f,g=im.shape[:2]

print(f,g)

#plt.imshow(d)

masked\_im=cv2.cvtColor(masked\_im, cv2.COLOR\_BGR2RGB)

# form a rectangle image along centre

image = cv2.rectangle(masked\_im, (27,H), (81,H),(255,0,0),-1)

image\_data = np.asarray(masked\_im)

count=0

for i in range(H-1,H):

for j in range(27,81):

print(image\_data[i][j])

if image\_data[i,j,0]==image\_data[i+1,j+1,0] and image\_data[i,j,1]==image\_data[i+1,j+1,1] and image\_data[i,j,2]==image\_data[i+1,j+1,2]:

count=count+0

else:

count=count+0.3

count=count/4

print('tdc=',count)

plt.imshow(image\_data)

image\_data=cv2.cvtColor(masked\_im, cv2.COLOR\_BGR2RGB)

aos=cv2.imwrite('tdc.png',image\_data)

"""

Checking using vibgyor segmentation

"""

import numpy as np

import PIL as Image

import cv2

import matplotlib.pyplot as plt

import sketcher

import cv2

image=a=cv2.imread('cornea circle.png')

plt.imshow(a)

a=cv2.cvtColor(a, cv2.COLOR\_BGR2RGB)

b=a.shape[2]

if b==3:

print('valid image')

else:

print('select a valid rgb image')

C = np.zeros(shape=image.shape)

if np.max(image) > 1:

GL = 255

else:

GL = 1

GL=255

f1, f2, f3 = (GL \* 1), (GL \* 0.6), (GL \* 0.68)

f4, f5, f6 = (GL \* 0), (GL \* 1), (GL \* 0.6)

def cfilter(image, f1, f2, f3, f4, f5, f6, m, flg):

C = np.zeros(shape=image.shape)

for i in range(image.shape[0]):

for j in range(image.shape[1]):

if flg == 0:

if f2 <= image[i, j, 0] <= f1 and f4 <= image[i, j, 1] <= f3 and f6 <= image[i, j, 2] <= f5 and image[i, j, m-1] == np.max(image[i, j]):

C[i, j] = image[i, j]

else:

C[i, j] = image[i, j, 0] \* 0.3 + image[

i, j, 1] \* 0.59 + image[i, j, 2] \* 0.11

else:

if f2 <= image[i, j, 0] <= f1 and f4 <= image[i, j, 1] <= f3 and f6 <= image[i, j, 2] <= f5:

C[i, j] = image[i, j]

else:

C[i, j] = image[i, j, 0] \* 0.3 + image[

i, j, 1] \* 0.59 + image[i, j, 2] \* 0.11

return C

C = cfilter(image, f1, f2, f3, f4, f5, f6, 3, 1)/GL

D=np.asarray(C)

print(D)

plt.imshow(D)