INTRODUCTION

The COVID-19 has introduced many challenges one of which is remote sensing. One of the prime characteristics of being infected is an elevation in the human temperature. In order to tackle this problem, it is convenient to apply remote skin temperature sensing thus inferring the person state.

REMOTE TEMP SENSING

For this specific application the goal is to place a thermal camera on the PC monitor and measure and record the person temperature. For a person seating 40cm – 74cm from a monitor (depending on its head size) it can be calculated that the required AOV is ~ 40° as we wish to overfill the person head to have sufficient pixels on the target. The best place to measure skin temperature is on a person forehead just above the eyebrows line since it is the most exposed part.

|  |  |
| --- | --- |
| A picture containing person, indoor, computer, computer  Description automatically generated | Diagram  Description automatically generated |
| Figure-1 : illustration of AOV | Figure-2 : illustration of head size |

At the time of writing this article there were a few alternatives for 32x24 pixel thermopiles such as Melexis , Excelitas and Heimann. The criteria for choosing the right HW were : complete optical module including >20x20pixel sensor resolution , thermal lens , temperature sensor for compensation, frame rate up to 16Hz , thermal precision ±1.5° and an IC (integrated circuit) that can handle an internal calibration parameters inside an internal memory as well as hold FPN (fixed pattern noise) compensation , drivers and physical interface to an external MCU, small form factor and support for Linux based systems.

After approaching the different companies and performing a benchmark I have decided to go with the mlx90640 board and the MLX90640ESF-BAB-000-SP thermal module.

<https://www.melexis.com/en/product/MLX90640/Far-Infrared-Thermal-Sensor-Array>

the mlx90640 has plenty of open source examples and information as well as complete python package for a fast integration on a raspberry-pi (USB or I2C).

HARDWARE

1. Raspberry-Pi 4B (8G RAM), but it can work on any board
2. RPI CASE with 12VDC FAN
3. Micro USB Power cable for the RPI (part of the RPI KIT)
4. Micro HDMI cable
5. RPI camera V2.1 (colored) + Flex cable
6. MLX90640-41\_evb\_2 evaluation board (Melexis)
7. USB3.0 cable (comes with the EVB KIT)
8. MLX90640ESF-BAB-000-SP 2D thermopile array (24x32/35°x55° AOV)

|  |
| --- |
| A picture containing indoor, table, sitting, computer  Description automatically generated |
| Figure-1: Setup |

METHOD

We will need to associate a person’s temperature and be able to both record it and notify if it exceeds a certain threshold.

Our target is to:

1. Detect that there is a person in the frame
   1. Segmentation – separating foreground and background
   2. Contour extraction – delate/erode and taking the largest contour
2. Locate his forehead
   1. Landmarks recognition (Eyes)
   2. Calculate Center of forehead
   3. Calculate forehead parallelogram
   4. Track the parallelogram (Kalman filter)
3. Measure its forehead skin temperature
   1. Mean value vs Exponential moving average (EMA)
4. Record the temperature during the day (CSV)
5. Display and Notify the system for any temperature exceeding

Other considerations:

1. Thermopile stabilization time – the thermopile technology requires ~3-5min for to overcome the temperature gradient created between the thermopile 2D array being heated actively and the external temperature.
2. Ambient temperature changes (Otsu) – while the former takes care of internal temperature change, this issue comes from the difference of temperature between the camera enclosure and the ambient temperature. In order to overcome this, we will use the “Otsu” algorithm to adaptively decide on the correct temperature threshold required to segment between the FG (person in the scene) and the BG.
3. Sensor fusion – we will use the thermal image, governing spectral response in the FIR ranging between 8-14 um and the visible spectrum ranging between 0.4 – 0.7 um and take the best from each sensor. For instance, we will use the thermal image to perform segmentation and the visible to perform face recognition. In order to overcome mechanical installation limitation, we will perform an affine transformation on the visible image and blend it with the thermal.

WORKFLOW

Pre-processing

1D-array

The thermal camera

outputs a 1-D array of size 32x24=768, we then need to:

1. Perform compensation – section 1 and 2 in the “Other considerations” paragraph as well as fix pattern noise and other noises corrections.
2. converting the values that are read in Celsius to digital values, between Tmin and Tmax (18° - 45°) as well as factor it by 255 in order to get values between 0 – 255.
3. Reshape the 1D array into 32x24 (WxH) 2D matrix readable by opencv
4. Perform LPF (Blurring) over the image
5. Resize the image from 32x24 to 320x240 so it will later fit the visible image resolution
6. Apply color map (JET) to give it red and blue (hot and cold respectively) heat bar

Segmentation

Pre-processing

1D-array

1. Perform segmentation:
   1. Find adaptive threshold, using “Otsu” algorithm
   2. Use the “Otsu” threshold and apply thresholding on the image to get BG = 0 and FG ≠ 255. This will act as a mask where the background is zero and the foreground is white.
   3. Apply “bitwise” multiplication “and” on the original (‘f’) image and get the foreground in color

It should be noted that I have used other ways to perform segmentation, such as contour search and also using ML clustering (k-means).

The visible camera

1. Resize the resolution from 640x480 to 320x240 in order to ease on the computational effort
2. We will need to flip (horizontally) the image
3. Apply a pre-calibrated affine transformation on each frame so as to get the overlay effect.
4. Apply blending between the segmented thermal image ‘g’ and the transformed visible image ‘c’ (can use cv2.add or cv2.addweighted where in the later we will use alpha blending parameter 0 < alpha < 1 to control the transparency.

Image alignment

1. Once the thermal and visible frames are aligned (matched at a pixel level) we can now perform “landmark-detection” and locate the left (37 - 42) and right (43 - 48) eyes of the person, calculate their centers and construct a parallelogram on the person forehead that will serve as the area on which we will measure his temperature. The idea behind the parallelogram is that when the person’s head is shifted to the sides and goes up and down the rectangle that is originally formed when the person is staring forward to the cameras will change into a parallelogram. NOTE: it is better to use pose estimation methodology to calculate the person forehead area, but for the mean while we shall use this rather simple method.

|  |
| --- |
| Chart  Description automatically generated |
| Figure-2: landmarks and parallelogram creation |

NOTE: It should be noted that the landmark detection usually consists of a “face-detection” algorithm from which we can apply landmarks-detection. In this work I used “Dlib” library as it is a very flexible and accurate algorithm that combine both face and landmarks detection.

1. We will apply a tracking algorithm on the parallelogram points so we will cover the case in which the person’s face is instantaneously lost
2. Out of the parallelogram area we will measure the maximum temperature
3. Perform exponential moving average (EMA) in order to sort of smoothen the stochastic readings
4. Finally, we will display and record the person’s temperature

Challenges:

1. One of the challenges that were discovered is the ability to recognize a person’s landmarks when he or she is wearing a mask.
2. Since the thermopile technology has a poor NETD with comparison to other bolometric sensors, the segmentation based on the thermal image will only work when there is a > 4°C difference between the object being measured and his background. Testing the algorithm on a hot day outside yielded bad performance. The conclusion is that when going with thermopile technology one should use it for indoor applications.
3. Another downside on the “skin measurement” devices is that when measuring a person’s skin while he is going through a large temperature differences the reading may very much be less accurate. The conclusion on this part is that reading skin temperature with thermopile is good for significant increase of body temperature (> 38°).

INSTALLATION

I have prepared an \*.img file that has all required libraries to run the module.

The file is located inside the toolkit library. A brief explanation of how to do this can be foung in the official raspberry pi website and I also used the below link:

<https://magpi.raspberrypi.org/articles/back-up-raspberry-pi>

VIRTUAL ENVIRONMENT

I recommend building a dedicated *virtual environment* where you can install all the necessary libraries into its “site-packages”.

There is a very good explanatory guide made by one of the gurus in computer vision:

<https://www.pyimagesearch.com/2018/09/19/pip-install-opencv/>

once you have established a working virtual environment you can install the mlx90640 libraries that can be cloned from the following Github URL:

<https://github.com/melexis-fir/mlx9064x-driver-py>

Graphical user interface, text, application

Description automatically generated

Open Terminal and enter “pip install mlx9064x-driver”



MODULE STRUCTURE

Building the “**RemoteSensing**” toolkit structure consists of five sub modules:

A picture containing text

Description automatically generated

**ImageAline** – it is an offline process that uses affine transformation to align the visible surface onto the thermal surface. One can read more in the following article

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html>

**Preprocessing** – performs image scaling, flip, color space conversions etc. on both visible and thermal frames.

**Segmentation** – this module is responsible to separate the foreground (person) from the background.

**Tracker** – uses tracking algorithms such as Kalman, particles filtering etc. to track the person forehead which is later used for temperature measurements

**Shape\_predictor\_68\_face\_landmarks.dat** – database used in “dlib” in order to locate a person face landmarks.

INSTALLATION

import cv2

import numpy as np

from imutils import face\_utils

from mlx.mlx90640 import Mlx9064x

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.neighbors import KernelDensity

import dlib

import argparse

import face\_recognition