UNIVERSITY OF THE PHILIPPINES

CATIFS via ICOTS: Towards Cheap and Accurate Thermal Imaging for Fever Screening Via Interconnected, COTS-Based Compute Devices with Optical and Thermal Sensors

Dee, Sam Johann Macandog

Bachelor of Science in Computer Engineering

Dela Cruz, Mariella Garcia

Bachelor of Science in Computer Engineering

Perez, Jose Francisco Adriano

Bachelor of Science in Computer Engineering

Ramirez, Kristian Dave Alcantara

Bachelor of Science in Computer Engineering

Yarcia, Drexzel Lopez

Bachelor of Science in Computer Engineering

Undergraduate Project Adviser:

Roel Ocampo, Ph.D.

Isabel Austria, Ph.D.

Rhandley Cajote, Ph.D.

Electrical and Electronics Engineering Institute

University of the Philippines Diliman

Undergraduate Project Reader:

Paul Jason Co, Ph.D.

Electrical and Electronics Engineering Institute

University of the Philippines Diliman

Date of Submission

15 July 2021

Permission is given for the following people to have access to this thesis:

Circle one or more concerns: I P C	
Available to the general public	Yes/No
Available only after consultation with author/thesis adviser	Yes/No
Available only to those bound by confidentiality agreement	Yes/No

Students' signature/s: Drexzel Yarcia

Signature/s of undergraduate project advisers:

University Permission Page

I hereby grant the University of the Philippines non-exclusive worldwide, royalty-free license to reproduce, publish, and public distribute copies of this work in any form subject to the provisions of applicable laws, the provisions of the UP IPR policy and any contractual obligations, as well as more specific permission marking on the Title Page.

Specifically I grant the following rights to the University:

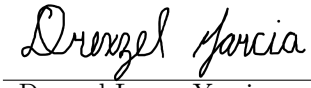
- to upload a copy of the work in the theses database of the college/school/institute/department and in any other databases available on the public internet;
- to publish the work in the college/school/institute/department journal, both in print and electronic or digital format and online; and
- to give open access to above-mentioned work, thus allowing "fair-use" of the work in accordance with the provisions of the Intellectual Property Code of the Philippines (Republic Act No. 8293), especially for teaching, scholarly, and research purposes.

Sam Johann Macandog Dee — July 15 2021 (Date)

Mariella Garcia Dela Cruz — July 15 2021 (Date)

Jose Francisco Adriano Perez — July 15 2021 (Date)

Kristian Dave Alcantara Ramirez — July 15 2021 (Date)



Drexzel Lopez Yarcia — July 15 2021 (Date)

Approval Sheet

In partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Engineering, this project entitled "CATIFS via ICOTS: Towards Cheap and Accurate Thermal Imaging for Fever Screening Via Interconnected, COTS-Based Compute Devices with Optical and Thermal Sensors", prepared and submitted by Sam Johann Macandog Dee, Mariella Garcia Dela Cruz, Jose Francisco Adriano Perez, Kristian Dave Alcantara Ramirez, and Drexzel Lopez Yarcia is hereby recommended for approval.

Roel Ocampo, Ph.D. Adviser	July 15 2021 Date
Isabel Austria, Ph.D. Adviser	July 15 2021 Date
Rhandley Cajote, Ph.D. Adviser	July 15 2021 Date

Accepted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Engineering.

Paul Jason Co, Ph.D. Panel Member	July 15 2021 Date
Michael Angelo Pedrasa, Ph.D. Director, Electrical and Electronics Engineering Institute	July 15 2021 Date

CATIFS via ICOTS: Towards Cheap and Accurate Thermal Imaging for Fever Screening Via Interconnected, COTS-Based Compute Devices with Optical and Thermal Sensors

Undergraduate Student Project

by

Dee, Sam Johann Macandog
2015-04381
B.S. Computer Engineering

Dela Cruz, Mariella Garcia
2015-03203
B.S. Computer Engineering

Hernandez, Lowell Andrew Cana
2015-05050
B.S. Computer Engineering

Perez, Jose Francisco Adriano
2015-14417
B.S. Computer Engineering

Ramirez, Kristian Dave Alcantara
2015-02910
B.S. Computer Engineering

Yarcia, Drexzel Lopez
2015-02176
B.S. Computer Engineering

Adviser:

Roel Ocampo, PhD
Isabel Austria, PhD
Rhandley Cajote, PhD
University of the Philippines, Diliman

April 2021

Abstract

CATIFS via ICOTS: Towards Cheap and Accurate Thermal Imaging for Fever Screening Via Interconnected, COTS-Based Compute Devices with Optical and Thermal Sensors

The need for thermal sensing devices experienced a surge as the need for fever screening and monitoring became essential and even mandatory worldwide due to the ongoing COVID-19 pandemic. Although there have been many thermal sensing devices, proprietary thermal sensing devices are mostly expensive and difficult to modify or improve, while open source thermal sensing applications still have a lot of potential for improvement in terms of accuracy, scalability, expandability, and manageability. Thus, this project developed an interconnected, open, programmable, and extensible thermal imaging platform which will provide a platform for a wide variety of applications that may lead to cost-effective and open source alternatives to thermal imaging devices.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of the Project

With the COVID-19 pandemic, the Philippine government recommended via the DTI and DOLE guidelines, that thermal screening be done upon entering establishments in an attempt to decrease the transmission of the virus [1]. However, thermal screening systems on their own are not capable of determining if someone definitively has COVID-19 because a person with COVID-19 may not have a fever. A COVID-19 test is still necessary to detect the disease. Thermal Screening, however may still lead to at least identifying someone who may potentially have COVID-19 and will need further screening[2].

The most popular option in thermal screening is the forehead thermometer as this instrument is the least invasive type of thermometer. However, forehead thermometers are prone to inaccuracy and inconsistency due to external factors, including drafts, wind, indoor heating, and direct sunlight, easily affecting this device. The reliance on an operator also presents and opportunity for human error. Along with this, it poses an increased risk of exposure to COVID-19 [3]. This makes the reliability of thermal screening questionable.

Thermal imagers provide a safer solution because even though it may still need an operator; it allows measurements from a farther distance that allows social distancing measures to be observed. It is also a more accurate solution because a thermometer is limited to one spot while a thermal imager captures an image. This negates the requirement of the thermometer for perfect conditions as one is able to choose the spot for the best reading [4]. However, there are two main drawbacks with thermal imagers. The first one is the price; these devices are really expensive, starting around $219(PHP 11,000) [5]. The second drawback is the software of these devices are closed and proprietary. Being closed and proprietary would hinder us from making any changes to the device such as modifications or upgrades. If there is any need for a new feature or a faster algorithm, we may need to buy another thermal camera which would be more money spent.

This project developed an open source thermal imaging system to realize the potential of existing thermal imaging devices by addressing the shortcomings of the aforementioned thermal screening devices. The proposed thermal imaging device would have the characteristics of being accurate, scalable, expandable, manageable, and cost-effective. Also, with an open source thermal imaging platform, this opens up the possibility for further improvements or applications in other fields.

1.2 Proposal Flow and Organization

In our review of related literature, standalone handheld thermometers, standalone thermal imagers, and interconnected digital thermometers and thermal imagers are explored in what they offer in terms of accuracy, scalability, expandability, manaageability, and cost-effectiveness. Chapter 3 discusses the problems posed by the solutions discussed in chapter 2 and states the objectives of this project in relation to addressing the gaps that have been noted. Chapter 4 goes through the design decisions and methods taken to achieve the set objectives and discusses the corresponding result of that action, as well as the possible improvements for future work.

Chapter 2

Review of Related Work

This chapter will present different existing studies and works that are related to the proposal. It is divided into five parts: Accuracy, Performance and Scalability, Feature Set, Manageability, and Cost-effectivness.

2.1 Accuracy

Accuracy in fever screening is the closeness of agreement between measured temperature readings and core body temperature [6]. This is essential for properly detecting fever. While there are many varieties of thermal imagers, not all are designed for taking accurate body temperatures. Most that do are usually expensive or closed-sourced. Fortunately, there are a number of ways to improve the accuracy of cheap thermal cameras.

2.1.1 Localization of Thermal Measurement

The best part of the face to measure temperature is the inner corner of the eye or canthus[7]. The temperature readings at the canthus are very close to the core body temperature, reducing calibration error [7]. However, the canthus region is very tiny and requires a high-resolution camera for it to be read.

An alternative is to take the temperature at the forehead, which has a $2.5°C$ differnce from actual body temperature [8]. Accounting for the temperature difference during recalibration gives a 96.7% accuracy[9]. The problem with taking temperatures on the forehead is that hair may sometimes block the forehead.

Another method is to take the full-face maximum temperature. Some tests proved that full-face maximum is more accurate than even the maximum forehead temperature [10].

2.1.2 Hardware-Based Approaches

One method of achieving higher accuracy is through calibration. Calibration is defined as the process of comparing a thermal reading with a reference that has been calibrated to a known parameter[11]. However, hardware calibration can only be done by the manufacturer due to the following reasons[12]:

- Requires multi-point calibration
- Using just one or two black bodies won't suffice
- No end-user access to non-volatile memory of camera electronics
- Needs non-uniformity calibration to measure and compensate for temperature drift

Another hardware based approach to achieving higher accuracy is through increasing the gain amplification in the integrated circuit (IC) of the thermal sensor. High gain amplification increases the accuracy of the thermal sensor, but reduces the thermal range as a trade-off [13]. Since our only concern is measuring human temperatures, we can increase the accuracy until thermal range is from $25 - 40°C$. However, just like calibration, increasing gain amplification can only be done by the manufacturer. Some manufacturers may accept custom request [13], but their services may be limited and expensive.

2.1.3 Software-Based Approaches

The advantage of improving accuracy via software is that we can implement it ourselves, given of course that the thermal imaging device is commercial off-the-shelf (COTS).

One approach is through mathematics or statistics. Averaging can give us an overview of how well our thermal sensor performs. However, outlier measurements may affect the mean. Taking the median is better for cases with outliers [14]. Another mathematical approach is through maximum likelihood estimation (MLE). Unlike averaging where we get a single value, MLE gives us a distribution (usually Gaussian) that best fits the data. Having a distribution allows us to make predictions. Studies have shown that using MLE for improving accuracy for wireless sensor networks (WSN) temperature measurements is better than conventional moving-average-filter [15].

A more advance approach is to use machine learning or neural networks. Neural network techniques such as face detection have been used to calibrate and maintain accurate temperature estimates [8, 9, 16]. Machine learning also helps improve accuracy by reducing human error and help ease record keeping.

2.2 Performance and Scalability

As differentiated from accuracy, we define the performance of a thermal imager as the rate at which it can indicate measurements and classify subjects over a certain period of time. This may be manifested as the time elapsed between the instance the subject is presented until either the measured temperature or condition (normal vs. elevated temperature) is reported or indicated. We call this elapsed time measurement as latency. [17]

Most of the standalone thermal imagers use face detection in order to localize temperature reading sites [8, 9, 16] but the introduction of these computationally expensive processes [18, 19] can slow down the thermal imager. With the help of built-in compute resources such as GPU, this can help improve the

system's latency. [8] Using multiple processing units can also speed up these processes but these are usually expensive. [20]

2.2.1 Throughput

Since a thermal screening may have to deal with multiple subjects, we are also interested in the throughput of a thermal imager. We define throughput as the number of subjects that are successfully scanned and processed over a period of time. [17] While this is closely related and perhaps dictated in a significant manner by the latency, there could be other external usage considerations or system factors not directly related to the measurement act itself, such as for instance the need for subjects to maintain safe distances from each other while measurement or classification is underway. [21, 22]

In standalone digital infrared thermometers, system-user interaction is one of the main problems since subjects still need to approach the thermometer in order to be scanned. This causes a bottleneck on the system's latency and these interactions also introduces risks to health. [21, 22] However, this process can be automated with the introduction of neural network techniques such as face detection. [8, 9]

Standalone thermal imagers encounter problems with partial occlusions and varying heights. [23] Being unable to adjust to varying heights means requiring subjects to adjust to the level of the thermal imager which slows down the process of taking the temperature. This can be overcome by cross referencing thermal and optical images to localize temperature reading sites on the images with the help of face detection. [8, 9] Problems with partial occlusions require subjects to remove pieces of clothing which cover the face which also slows down the system. Data sets of masked faces are already available for training face detection models which can help bypass this problem. [24, 25]

2.2.2 Scalability

Finally, we define scalability as an ability to adapt to an increase in the number of subjects by enhancing or improving performance, that is, by reducing latency or increasing throughput. [26] This is desirable for situations where there might be an expected increase or even surge in subjects. It may be possible for example for a thermal measurement and screening system to offload computational tasks to a more powerful device or component such as a server in order to reduce latency. This can be done by interconnected digital thermometers and imagers. [16, 27] However, this type of thermal imagers are limited by their link speeds. Using these interconnected devices as processing nodes or storage nodes can help reduce the effects of slow link speeds since communication between devices is minimized and compartmentalized. Processing nodes such as GPU-enabled servers or GPU devices can run computationally expensive processes such as face detection while storage nodes can be used for processing the image streams. [20]

Another possible way to improve scalablity is for a system to deploy or integrate functionality that would allow it to reduce inter-subject measurement or classification. For example, throughput can be increased by performing measurements on several subjects simultaneously without having them physically queue up. [8, 9, 16] This reduces the system-user interaction and the time to process crowds or a sudden influx of subjects.

2.3 Feature Set

Commercial thermal screening systems are applications which are used to detect elevated skin temperatures using thermal cameras. These systems contain features to perform their basic functionality of providing an accurate measurement and classification. Commercial systems utilize face detection algorithms and calibration in response to the average skin temperature in order to detect individuals with elevated skin temperatures[28]. Systems utilize face detection algorithms to estimate the distance of the subject from the camera. If the subject is too far, the size of the face will be too small. To make sure the subject is in a correct position, the system will only measure temperatures when a person's face covers a specified percentage of the region-of-interest. Face detection algorithms is also used to reduce false alarms from other heat sources. Thermal screening systems calculate a moving average based on a specified number of samples. Only temperatures measured that are not classified as an elevated temperature are included in the average calculation. People with elevated temperatures do not contribute to the calculation of the average temperature.

Other key features that are present on commercial solutions are related to supporting remote operation for safety reasons[29]. By allowing an operator to use the system from a separate location, the system can be used on public spaces such as airports, office buildings, factories, train terminals, concerts or other events, and other locations where the system can be set up on entrances. These features include, controlling the camera remotely, an alarm system for when a person with an elevated temperature is detected, viewing the thermal images in live mode, and creating image snapshots on detection. Most systems contain these features but other features outside of the basic functionality of thermal screening can also be present. These features include face recognition or face mask detection. These features are primarily used for authentication or for attendance purposes.

2.4 Manageability

In this section we evaluate existing thermal screening devices and platforms in terms of manageability, that is, the ability for a device or system to be managed or administered [30]. When part of a system, this may refer to enterprise-wide administration of distributed components, called systems management [31]. If we consider the wide spectrum of thermal screening needs ranging from one applied to a single control or observation point, such as the entrance of a single building or establishment, all the way to large-scale monitoring and health surveillance such as in a mall or an airport with several ingress and egress points and multiple layers of access control, then we can apply the concepts and principles of systems management to the issue of managing thermal screening systems.

The concepts behind systems management are strongly influenced by network management initiatives in telecommunications [31]. A popular management model in telecommunication systems is the FCAPS model, which stands for fault, configuration, accounting, performance and security management model [32]. Therefore, as may be applied to the review and evaluation of the state-of-the-art in thermal screening systems, manageability may be considered along these functional areas:

- Fault Detection - is it possible to detect when a temperature measurement point is not working properly or according to its specified or expected behavior, functionality and accuracy?
- Configuration - is it possible to systematically and dynamically configure temperature measurement devices, perhaps remotely and conveniently from a single management terminal, for example to modify alarm set points, processing algorithms, account for changes in the ambient environment, or even

to update firmware?

- Accounting - can usage history statistics be systematically gathered, possibly to anticipate or predict failure, optimize use and deployment, or plan for future growth and upgrades?
- Performance - is it possible to systematically monitor and evaluate performance, possibly to proactively plan or adapt the deployment of resources, or to check whether upgrades or enhancements are needed? Does the system continue to gather, generate, store or deliver information in a timely manner?
- Security - are all the devices in the system secure, and considering the nature of the data being collected and processed, does the system continue to protect privacy and confidentiality?

Based on these concepts, we now evaluate available and proposed (i.e. experimental) thermal screening platforms in terms of manageability. These platforms are roughly grouped as follows:

- Standalone handheld digital infrared thermometers
- Standalone thermal imagers
- Interconnected digital thermometers and imagers

2.4.1 Fault Detection Management

Standalone handheld digital infrared thermometers have arguably been among the most visibly used devices in the ongoing pandemic, particularly at entrances to small buildings and facilities. Examples of such devices include LPOW thermometer, iHealth No-Touch Forehead Thermometer, and Candy-Care[33][34][35]. Typically, these devices are manually operated, and since these are not connected to an external monitoring system, fault detection (i.e. device failure) can only be done manually as well, and only when detected by the operator. An operator might not be able to detect some faults such as erroneous measurements or drift from calibration. Furthermore, there will have to be a system of manual fault reporting in place if these are deployed in large numbers or in some distributed fashion.

Standalone thermal imagers do not differ significantly in terms of fault management as compared to the standalone infrared thermometers previously discussed. However, the addition of a new functional component – the display screen – perhaps highlights an even greater need for fault management. The additional component is a potential source or location for failure. The addition of a display screen does not confer any additional ability to automatically detect drift or inaccuracy, unless the screen itself is used as part of a manual recalibration or 'sanity check' process against a black body or some reliable standard [11]. Since these devices are by definition stand-alone, any fault detection features or functions will have to be done by the operator, and there can be no central automated fault monitoring nor reporting possible.

Interconnected digital thermometers and imagers are solutions that rely on a system of device be it combinations of hardware, software, or both. The companies giving solutions do not market the fault detection capabilities but they have features that may fall under this category such as web-based tool support that will help in troubleshooting the device. Interconnected digital thermometers and imagers also operate on a full OS or is at least connected to one.This allows reconfigurable devices to develop for or install with fault detection if they do not have one. An example is CSA LLC and SZZT Electronics Co. Ltd.'s machine which allows API docking which would allow fault detection software [36].

2.4.2 Configuration Management

Low-cost standalone handheld digital infrared thermometers are typically not configurable, or only have minimally configurable features such as turning on alarms for high or low temperatures, and changing temperature units to Celsius or Fahrenheit which the models stated above have. Other more expensive models such as FLUKE-572-2 CAL allow for emissivity parameters to be manually configured by the operator. In general however, these somewhat popular low-cost thermometers have little to no configurability, and do not allow for centralized configuration management when used in large numbers or in a geographically distributed setting.

The addition of a color display on thermal imagers also adds another configurable parameter to the standalone thermal imagers: the color-coding scheme or scale corresponding to temperature readings. While this offers some flexibility, it may also be a double-edged sword because the interpretation of the color-coding scheme may now become a subjective and operator-specific matter, and thus a sudden change in operator may result in erroneous interpretation of image readings. Since these devices are standalone, there are typically no mechanisms to centrally and remotely standardize the color coding schemes nor 'lock-in' those schemes, i.e. prevent operators from modifying or tampering with standardized presets.

Interconnected digital thermometers and imagers with the vast options available, promote more features. With more features for these devices, comes with more configurable options. This may come as a double-edged sword as having more configurations would require more resources to manage. By having connected systems, having an adjustable mount is now possible as demonstrated by the Senda Electronics kiosk [36]. Most of these devices allow a more adjustable alarm threshold, which may be limited in the previous classifications. Being interconnected also allows these devices to be integrated to company systems.

2.4.3 Accounting Management

Most of these low-cost handheld thermometers do not store usage statistics nor provide facilities for such statistics to be shared or archived. An exception might be devices like the FLUKE-572-2 CAL, which allows readings to be downloaded through its built-in USB interface. While this sets it apart from other similar products in its class, this makes gathering statistics across several scanning points a time-consuming and effort-intensive task, and does not allow statistics to be collected in real-time.

Most of standalone thermal imagers offer storage through either onboard storage or a microSD card with some exceptions like the FLIR TG56[**FLIR TG56**] . These allow the devices to keep recordings of thermal images and even some models allowing visible light images. There are also devices that include bluetooth and/or wifi such as the FLIR E4 which would allow real-time collecting and analysis of readings but with the definition of being standalone these devices themselves cannot keep (remotely) nor analyze the readings themselves on their own without the use of other devices[**FLIR e4**].

Interconnected digital thermometers and imagers mainly record temperature, but they can also record facial profiles and information set by the user. These are stored by the devices either on-board, as demonstrated by LamasaTech Kiosk, or in a server, like the IntraEdge Janus Kiosk [37][38].

2.4.4 Performance Management

Given the typical usage of standalone handheld thermometers, that is, for places where the number of subjects to be screened per unit time is expectedly low, performance is usually not a primary consideration. Therefore these also do not provide facilities for performance to be systematically monitored nor modified.

Standalone thermal imagers do not differ significantly in terms of performance management as compared to the standalone infrared thermometers previously discussed as they do not provide facilities for performance to be systematically monitored nor modified.

Interconnected digital thermometers and imagers are able to allow their devices to perform optimally. The BOXER-8120AI with NVIDIA Jetson TX2 uses AI and deep learning ensure that the measurements it performs accurately by taking into account external interference. Devices like the Palmer Group Kiosk simply prompts the user with desired conditions for the most accurate measurement. [36] Applications that also track resource management are easily available for these devices as they run on an Operating System.

2.4.5 Security Management

Finally, since most standalone handheld thermometers do not identify, store nor transmit personal information, there is usually very little concern for device-based security management. Furthermore, due to the standalone nature of these devices they are less vulnerable to network-based attacks or compromise. It is not currently known whether devices with built-in USB interfaces such as FLUKE-572-2 CAL have known vulnerabilities, although this is quite plausible, given the variety of USB-based attacks that are known [39].

As with standalone handheld thermometers, standalone handheld thermal imagers, do not identify, store nor transmit personal information, so there is also very little concern for device-based security management. However, some models can capture visible light photos/videos which may invade other people's privacy though the device will not be able to identify the persons involved as these devices are not made for facial recognition. There are also models with Bluetooth and/or WiFi such as the Fluke TiS20 and FLIR E5-XT that may pose vulnerabilities due to WiFi and Bluetooth attacks [40][41].

Interconnected digital thermometers and imagers provide multiple security management features. Some offer tamper protection as well as alarms when strangers are detected with the facial recognition. These devices also provide secure servers for data management [42].

2.5 Cost-effectiveness

The cost of a thermal screening device or system is an important consideration in its adoption and deployment, especially in relation to its intended use. Although absolute cost may be a major factor, users typically calibrate these with respect to the other criteria mentioned in the previous sections such as accuracy, performance, features, and functionality. For example, a small private office or building might be content with a simple handheld infrared thermometer, and would thus be unwilling to spend more to obtain added features or functionality beyond the basic needs of temperature measurement. A medical facility on the other might have stringent requirements on accuracy, and may therefore be willing to spend

more to meet such requirements. In contrast, a large facility such as an airport might place a premium on performance and throughput and may even be in need for added functionality such as measurement recording and subject tracking.

Therefore, while absolute cost may be a consideration, this is usually with the context of other characteristics and metrics. It is therefore useful to consider cost-effectiveness, which also takes into account these other characteristics and represents a perception of "value for money." In this section, we take a qualitative and very broad look at costs and the characteristics, features and functionalities that are typically available at certain price points or ranges. More importantly, we also examine whether it is possible to dynamically improve or achieve added value or enhance cost-effectiveness within existing platforms.

Typical Characteristics, Features and Functionalities of Devices at Different Price Points

For the purposes of this discussion, we continue to make use of the previously defined classifications of thermal imagers: SDIT, STI, and IDTI. However, we will also employ additional categories based on price point unto which thermal imaging devices will be further classified. These price points are defined in such a way that deviations in characteristics, features and functionalities of the devices within the range are comparable. With this, we define Low-Cost Devices to be thermal imaging devices below 20,000PHP while High-End Devices are thermal imagers priced at above 20,000PHP and above.

2.5.1 High-End Devices

High-end thermal imagers are typically STIs distributed and manufactured by well-known and established companies in the thermal imaging industry. Leading these companies in the field of thermal imagery are Flir Systems and Fluke Corporation and will thus be the primary subjects of discussion in this section[43]. FLIR Systems is the world leader in the design, manufacture, and marketing of thermal imaging infrared cameras[44]. Their products are known to be high performance yet reasonably priced units [43]. On the other hand, Fluke has been in the industry for over 70 years and is known for excellence through manufacturing high quality products[43]. While the quality of devices provided in this price range can go without question, it is worth exploring how these products fare in cost-effectiveness; particularly, in terms of upgradability and manageability.

Upgradability Upgradability of systems are of significant importance due to how fast-paced innovations are developed in our time. Upgradability entails adaptability which allows for specialization to certain applications; flexibility which allows for repurposing of the devices and future-proofing; and long term use, since being capable of upgrades can delay the obsolescence of the product. While high-end devices do allow for upgrades, they often come at an added cost. As an example, FLIR charges additionally for their own remote access software at almost 30,000PHP despite their cheapest compatible camera with this software coming at a price of almost 240,000PHP. Thus, cost-effectiveness in terms of upgradability still remains a point of improvement for high-end devices.

Manageability One of the characteristics that manageability encompasses is the repairability and replaceability of parts. To provide this, manufacturers typically offer warranty to their products. But, it must be noted that it is typically only offered for free for a limited to a few years and availing of services for repairs

and parts replacement beyond this period requires paying a fee[45]. Warranties also often do not cover all services and certain conditions have to be met to continue availing of the service[45]. One of which is restricting repairs and replacement services solely to manufacturers[46]. Thus, users do not have the option to avail of possibly more affordable third-party repair services and part replacement. Having said all this, it can be concluded that high-end devices are not the best choice for cost-effective solutions in terms of manageability.

2.5.2 Low-cost Devices

Commercial SDITs and hobbyist-grade IDTIs are what generally fall under this category. However, there are also commercially available low-cost STIs being offered in the market. As such, this section shall be subdivided accordingly.

2.5.2.1 Commercial Standalone Thermal Imagers

In this section, we cite a study that performed a general overview of the capabilities of low-cost cameras in comparison to a high-end model and sensor characterization to discuss the cost-effectiveness of commercial STIs[47]. The results of the said study showed significant dependence of low-cost commercial STIs on room temperature. In other words, these devices were found to be inaccurate. Recommendations for improvement are temperature compensation algorithms to be employed in applications in which a non-thermally controlled environment is considered and the exploration of the benefits of an external temperature-homogeneous reference object in low-cost sensors. Thus, low-cost commercial STIs can still be further developed to provide even better value for money.

2.5.2.2 Standalone Digital Infrared Thermometers

As mentioned in a previous section, these devices have arguably been among the most visibly used devices in the ongoing pandemic. One of the reasons for this is their affordable price in the range of only around 200PHP to 7000PHP[48]. However, while the affordability of these products are enticing, affordable is not always tantamount to cost-effective as affordability often comes at the expense of desirable characteristics being subpar. For the case of Low-cost SDITs, the diminished quality is accuracy similar to low-cost STIs.

Accuracy Research has shown that, when used correctly, infrared or no-contact thermometers are just as accurate as oral or rectal thermometers, the human operator requirement makes it susceptible to errors due to mishandling [49]. As an example, a medical expert can be quoted saying that these devices are ”notoriously inaccurate” due to his experience of being tested frequently by infrared thermometers, and having results suggesting he was dying of hypothermia[50]. In spite of this, there is no way to work around these limitations due to the inextensibility of these devices. Considering this, it is worth asking if this solution sufficiently grants the best value for money since its affordability comes at the cost of accuracy.

2.5.2.3 Interconnected Digital Thermometers and Imagers

Because of the limitations mentioned in the previous discussions in this section, research and development towards cost-effectiveness has been an area of interest in this field. However, breakthroughs are still often limited to certain applications only. Some examples are in volcanology, electrical fault monitoring, and monitoring the condition of mice and cannot be used on fever screening for humans due to how they are calibrated. While there are some that did focus on fever screening applications, the field still remains worthy of exploration since these systems are not readily available and widely distributed. Furthermore, there are also areas of improvement wherein the capabilities of such systems are more greatly utilized.

Versatility Temperature measurements can be categorized into those that do not require accuracy, those that do require accuracy, and those where there is uncertainty about the accuracy requirement[51]. Electrical fault monitoring and volcanology applications fall into the first category[52][53]. For electrical fault monitoring, it is only a matter of detecting high temperatures and not measuring them precisely, to determine if there are faults[52]. Thus, precisely accurate measurements are not needed. Similarly in volcanology, the usual point of interest is the trend of temperature to determine volcanic activity; thereby also not needing exact measurements since what is being considered is the trend of measurements[54]. In the case that there is a need to determine the range of temperature values for a volcano to be considered stable, precise measurements are also not needed since they only need an estimated range and minute temperature differences in the tenths scale seem negligible compared to the temperature measurements being taken in the thousands scale of this high temperature environment. Having said all this, low-cost IDTIs built for these kinds of applications cannot be used on thermal screening for humans where the temperature ranges in consideration are in the ranges of 30-40 degrees and minute differences in the tenths range can mean the difference between a febrile and afebrile individual. On the other hand, a closer application to fever screening for which an IDTI was developed for is mice condition monitoring. While this application does address monitoring of living species, it is specific to small creatures and close proximity monitoring and cannot thus be used for the purposes of our study[55]. Nonetheless, the utilization interconnectivity capabilities of IDTIs is commendable. This further proves the point that there is a need for flexibility as to which developed devices can be used for.

Interconnectivity, Programmability and Extensibility Through the course of our research, the solutions for this specific application we have covered thus far have not taken advantage of the interconnectivity capability of IDTIs despite their promising systems[16][56][57][58][59]. Their developed systems tend to be designed for standalone use. While most systems do utilize this, it is only done so by having a system comprising of a server and an IDTI node alone[56][58]. On the other hand, the programmability aspect, as well, is typically not fully utilized. As an example, a graphical user interface could be beneficial to users as this provides ease of use and yet this feature has not been added to the systems we were able to scope. It is noteworthy, however, that addition of face detection by utilizing the programmability aspect has already been explored[16]. Lastly, extensibility is the capability of IDTIs that is commonly made use of. However, this is mostly done by interfacing monitors only[56][59]. Nonetheless, such aspect of IDTIs may still be worthy of exploration by considering other hardware beyond monitors similar to that suggested by [47].

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

Commercial thermal imaging solutions used for fever screening are closed source and proprietary, making these solutions expensive and difficult to improve. In turn, these solutions are inaccessible and pose a hindrance to developing and exploring new techniques. Thus, additional and often greater expenses are imposed on those who want to upgrade these devices. While there have been initiatives in providing open source solutions for the applications of thermal imagery, some still use thermal cameras such as those distributed by FLIR thereby still constraining these systems to come at an expensive price. Because of this, commercially off the shelf (COTS)-based solutions with interfaced low resolution thermal sensors have also been under exploration. However, solutions under this category still remains to be limited to specific applications only. These solutions also pose limitations in resolution, accuracy, and potential in scalability as well as in providing user-friendly systems. Thus, we are to develop and demonstrate an interconnected, open, programmable, and extensible thermal imaging platform consisting of interconnected, COTS compute devices equipped with optical and thermal sensors. We aim to demonstrate that we can provide an easy-to-use, scalable, and cost-effective thermal camera that may lead to cost-effective, open source alternatives for thermal imaging for a wide variety of applications such as fever screening.

3.2 Objectives

Our project aims to develop an interconnected, open, programmable, and extensible thermal imaging platform that may potentially provide a thermal camera for a wide variety of applications leading to cost-effective, open source alternatives for thermal imaging.

In order to accomplish this, the goals of the development of the thermal imaging system will be with respect to the following:

- Accuracy
  - Improve accuracy of thermal reading using software techniques
  - Integrate a blackbody reference circuit into the system

  - Calibrate accuracy of thermal camera by using blackbody reference
- Scalability
  - Speedup by using built-in compute resources
  - Speedup by offloading tasks to server
  - Demonstrate ability to process multiple subjects
- Feature Set
  - Integrate iris scanner and thermal imager
  - Identify subjects either on the device or through a separate server for contact tracing
- Expandability
  - Create Virtual Machines/Containers that are able to run specific tasks
  - Chain Virtual Machines/Containers Services
  - Compare which type of virtualization would be the best option
- Manageability
  - Establish a network of devices in the system and enable manageability via network control
  - Create a modular network for quick push of updates
  - Demonstrate Ease of Fault Detection, Configuration, and Performance Management
- Cost-effectiveness
  - Use open and programmable platforms
  - Use of affordable components

Chapter 4

Methodology, Results, and Conclusion

This chapter discusses the details of how the objectives of the project were met. It focuses on showing our custom implementation of a thermal imaging platform in order to meet the desired characteristics defined in the introduction, as well as the results and conclusions drawn from the design decisions made in conceptualizing this system.

4.1 Manageability

To recall, this project demonstrates the advantage of employing an interconnected, open, programmable, and extensible thermal imaging platform. Thus, presented hereon is its benefits, in particular, in the terms of manageability.

As stated in the introduction of this document, manageability is one of the desired characteristics of thermal imagers. This is because manageability entails that the system has the ability to monitor and configure several sensors and even several sites safely and conveniently. This aspect of the project developed one means to achieve this by focusing on fault, configuration, and performance management for as discussed in chapter 2, these are some of the functional areas that may be applied to the review and evaluation of the state-of-the-art in thermal screening systems in terms of manageability based on the FCAPS model.

To reiterate the discussion in chapter 2 briefly, we summarize here that the gaps in existing solutions in relation to the functional areas aforementioned here include manual detection of faults, having little to no configurability, and performance being a non-priority. Manual detection of faults may lead to delayed detection of faults or not detecting faults at all. In the case that faults are detected, manual fault reporting should be done when these are deployed in large numbers or in some distributed fashion which is inefficient. Meanwhile, existing solutions having little to no configurability does not allow for centralized configuration management which is also inefficient when used in a geographically distributed setting with a large number of devices similar to fault reporting. An example of this is when setting standardized presets, every device need to be visited to be configured one by one thereby proving tedious in a geographical distribute setting with a large number of devices. On the other hand, solutions that do support configurability only have minimally configurable features such as turning on detection of high or low temperatures and changing temperature units to be Celsius or Fahrenheit. For thermal imagers that have display units,

there are solutions that have configurable color coding schemes. This feature of configurability, however, introduces the possibility of operators tampering with the settings thereby allowing each station to have differing settings. Thus, a sudden change in operator may result in erroneous interpretation of image readings since the settings are not standardized. Lastly, performance not being a primary consideration for existing solutions due to the setting of having subjects to be screened per unit time being expectedly low, provides more room for error.

To address these points of improvement in line with the fault, configuration and performance management aspects of the FCAPS model, this part of the project provides a management web application that may be used for interconnected thermal platforms. We solved the problem of manual fault detection by implementing automatic fault detection in the form of fault indicators in the web app, to improve fault management. On the other hand, to solve the issue with regards to configuration management, our system provides a means to configure all nodes centrally with standardized presets that prevents modification or tampering of these settings outside of the system. Lastly, this web app has facilities for uptime and performance to be systematically monitored to improve performance management. The methods employed for the implementation of this system is discussed in more detail in the following subsection.

System Design and Implementation

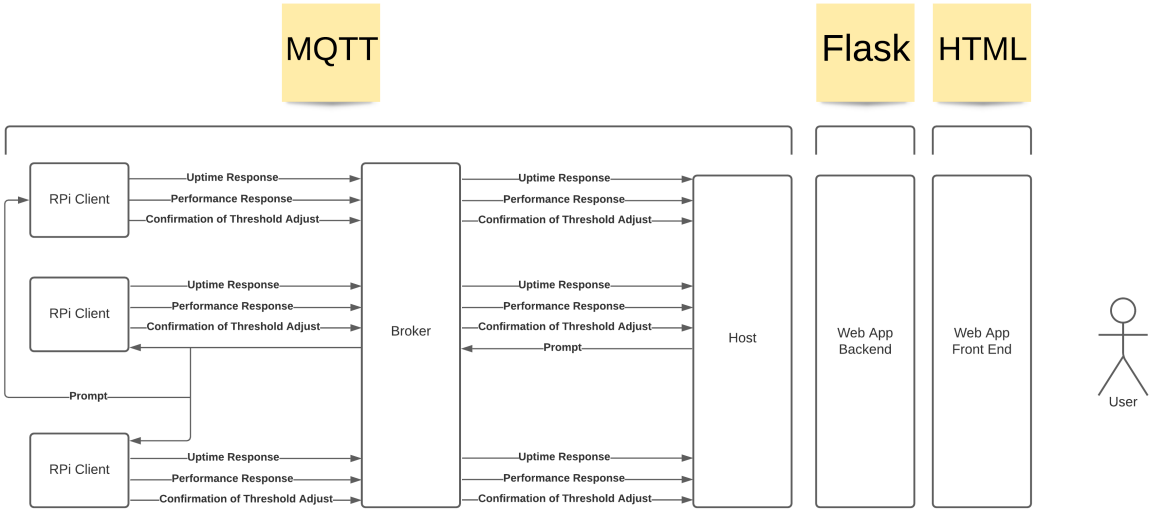


Figure 4.1: Management Module System Diagram

Shown in the figure above is the system diagram of the management module. The implementation of the system primarily made use of the open source projects, Flask and MQTT. Flask is a python based web framework that allowed us to create the web app itself while MQTT is a publish/subscribe messaging transport which we used for the communication among the host and devices being monitored[60][61]. The MQTT protocol consists of two hosts, clients and a broker. Clients may either be publishers or subscribers while the broker is the one that manages sending the messages published by the publisher to their respective subscribers. A client may be a publisher and subscriber at the same time. The broker handles this using MQTT topics. MQTT topics are the identifiers of the channels that clients publish or subscribe to[62].

The devices monitored for this experiment are virtual machines used to emulate a Raspberry Pi(RPi). Each RPi was designed to be an MQTT client that has 5 dedicated MQTT topics to which it publishes to and the host is subscribed to. All RPis have one shared topic to which they are subscribed to in which the value they will change their threshold variable into is published by the host. These topics are as follows:

1. Performance Monitoring
    - CPU Usage Topic - topic in which the device being monitored publishes their current CPU load
    - Memory Usage Topic - topic in which the device being monitored publishes their current memory usage
2. Uptime Monitoring
    - Connectivity Topic - topic in which the device being monitored publishes their status of connectivity. The devices publish into this topic upon connection to the broker to let the host know that they are connected and upon disconnection to let the host know that they have been disconnected.
3. Threshold Adjustment
    - New Value for Threshold Topic - topic in which the host publishes to the clients the value of the fever classification threshold they should follow upon receiving the message.
    - Change of Threshold Response - topic in which the devices publish what their previous value of fever classification threshold was and their current fever classification threshold after the host published a value in the ”New Value for Threshold” topic.

Fault Detection

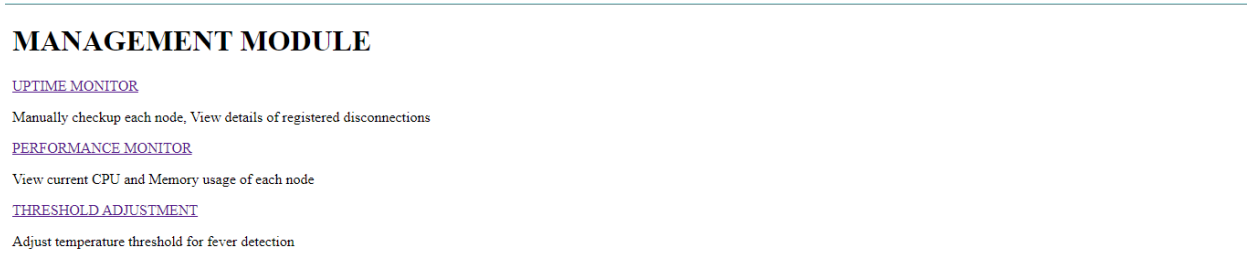


Figure 4.2: Management Module

Shown in the figure above is the web application integrated with the MQTT client that serves as the host. The fault detection feature is primarily where the web application proved useful. The web app provided a way to monitor the devices centrally allowing us to have some kind of notification system by incorporating indicators of faults in the web app. Each of the three submodules to be discussed in the following subsections have this fault detection indicator. For the ”Performance Monitoring” submodule, an indicator of whether the CPU Load or Memory usage is high or low is displayed beside the value of the current percent usage of CPU and Memory. Next, for the ”Uptime Monitor,” having two ways to detect connectivity can help determine the kind of disconnection that a node underwent. The packet drop rate is also provided in this submodule which can also help in diagnosing the problem. Lastly, the threshold adjustment module presents the current value of the nodes' threshold variable as well as its previous value. This data is sent from a node to the host. This way we can ensure that the nodes did indeed update their thresholds. These three submodules are discussed in more detail in the proceeding sections.

Performance Monitor

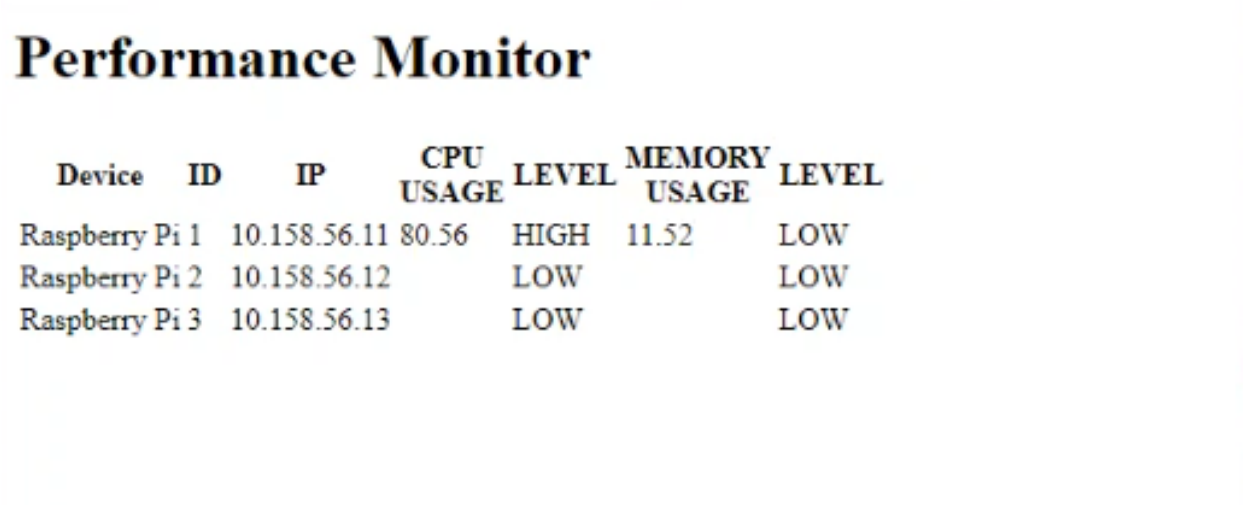


Figure 4.3: Performance Monitoring Submodule

Shown in the figure above is the performance monitoring submodule. For the performance monitoring feature, each node sends CPU load and memory usage to the MQTT broker which then forwards this to the host with an integrated web application. This data is then displayed in the submodule titled ”Performance Monitor” and stored locally on the host and processed for the indicators to reflect whether the usage is high or low.

Uptime Monitor

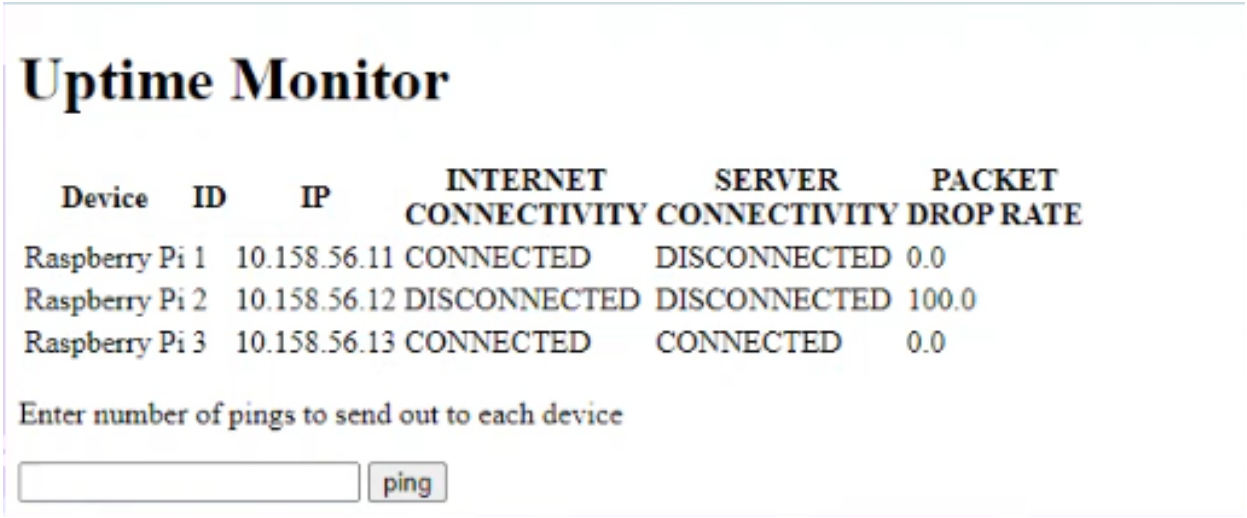


Figure 4.4: Uptime Monitoring Submodule

On the other hand, MQTT has a built in feature called last will and testament which allows disconnecting MQTT clients to send out a message if they were disconnected disgracefully[63]. Thus, this is what the project utilized to determine if each node was connected or not. However, this has a limitation of only detecting disconnections from the MQTT broker. To fill this gap, an added feature to the system is to allow the user to do ping sweeps to check if the node has been disconnected from the internet as well. Ping sweeps are done via an Internet Control Message Protocol (ICMP) echo request. Moreover, ping sweeps employ multiple ICMP echo packets being sent simultaneously to several hosts. All live hosts will then ping back [64]. This way, ping sweeps discover which IP addresses are active or live on the network. The information regarding node uptime is displayed in the submodule titled ”Uptime Monitor.”

Configuration Management

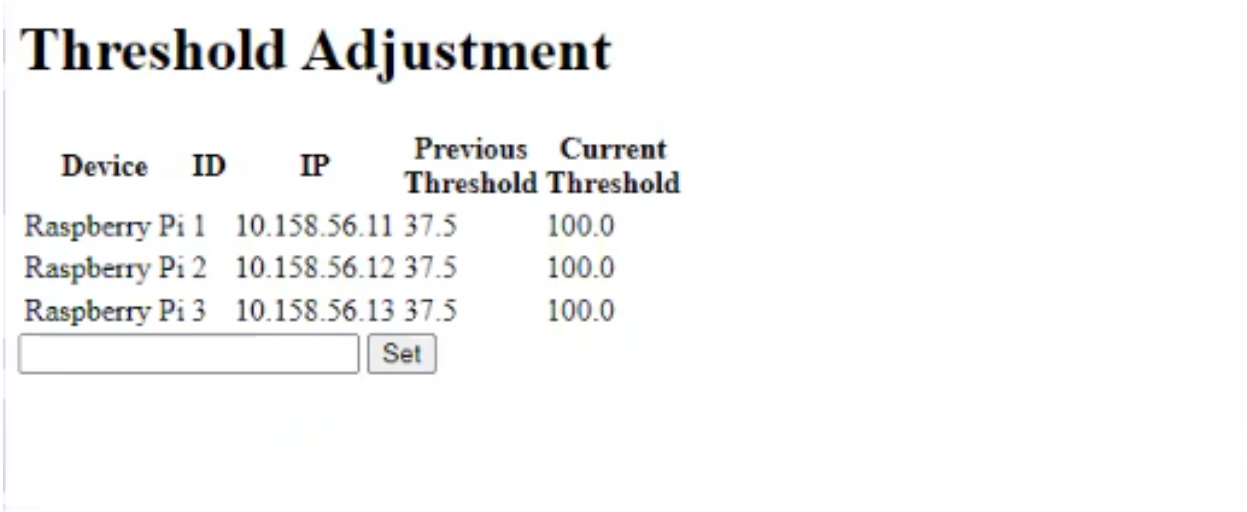


Figure 4.5: Threshold Adjustment Submodule

For configuration management, the system utilized an MQTT topic dedicated for the host to send the value into which the nodes will change their fever classification threshold into. In turn, the nodes are subscribed to this topic to receive this value. Although it may seem straightforward, this illustrates that the system can remotely configure the nodes which may also go beyond just changing a simple variable. Users can make use of this feature and enter the value to change the threshold into in the ”Threshold Adjustment” submodule of the web app.

Evaluation, Results, and Conclusion

Interconnectivity

For the interconnectivity of devices, the publish and subscribe model application layer protocol, MQTT, allowed the server to communicate to the system nodes at virtually the same time thus removing the complexity of having to communicate to these clients individually. Being built on top of TCP, MQTT granted no need for the nodes nor the host to keep track of IP addresses of intended message receivers. However, the need to keep track of who sent them was still necessary.

Station Uptime Monitoring

The built-in Last Will and Testament (LWT) feature was useful for the purposes of this station uptime monitoring. This, however, was insufficient as it only was applicable to disconnections that were due to network failure or termination of the program[63]. For exceptions such as keyboard interrupts, this message was not sent as the MQTT context manager cleans the disconnection for these cases[65]. Thus, this was handled by having the client publish in the on disconnect callback so that every time a client disconnects, they will explicitly notify the server. Although the incorporation of recurring automated system checkup through ping scan and cron was initially proposed, this was instead replaced by allowing

the user to ping upon their discretion so as to check if the LWT based connectivity indicator is only showing disconnection to the broker or network disconnection [64][66].

Performance Management

For performance data collection, memory and CPU usage was sent from node to host indefinitely instead of having MQTT topics for host prompts and client replies. For the alert for issues, on the other hand, this was simplified to fault indicators. Evaluation of the system was done by using the stress-ng tool on the nodes to induce CPU and Memory usage stress and checking if this was reflected in the module[67]. These tests indeed verified that the feature was working as intended.

Configuration Management

For configuration management, the implemented dynamic fever threshold adjustment made use of an asynchronous approach when changing the value of the fever classification threshold on the Raspberry Pi as this was the innate configuration of MQTT. We were not able to test this, however, in adjusting the thresholds across all stations while running an actual fever detection system as there has been no chance to integrate this with other parts of the project.

Web server with User Interface

A web application was put in place of a web server to allow central monitoring and configuration. A user interface was provided for further ease in the use of the management system. A web server was not employed due to the limitations of the resources provided such as having a setting of being able to deploy and test the syste within a local area network only. Three implementations were considered when designing the integration of the web app to the MQTT host. These approaches are the following:

1. An implementation allowing for a modular architecture which splits the webapp and MQTT and having the web app and mqtt client communicate using fifo files. However, it limits access to the system to a single device since the web app and MQTT host need to be in the same machine when communicating via fifo files. It also comes with operating system synchronization overhead and implementation complexity overhead

2. An implementation having two separate programs for the webapp and the MQTT host and have them communicate via websockets. The host will then reside within the same local network as the devices being monitored while the webapp can be accessed anywhere beyond the network. this introduces an additional and unnecessary communication overhead between the web app and mqtt host, and an additional implementation complexity overhead for to accommodate the use of websockets.

3. The option adapted by this project due to the simplicity of implementation and least resource overhead, the integration of the MQTT client and web app into a single application. Should the web application with MQTT host be preferred to be accessed from beyond the local network, packet transport can easily be set to websockets as this is a built in feature of MQTT. However, a web server may be needed for the storage of the performance data.

Conclusions and Future Work

The management web application was able to achieve its goal of providing fault, configuration, and performance management by providing automated detection of faults to aid fault management, providing a means to configure all nodes centrally with standardized presets which prevents modification or tampering of these settings outside of the system to improve configuration management, and storing logs of CPU and Memory Usage as well as identifying performance or non-performance through the uptime monitor which solves the problem in performance management.

This is comparable to the existing solutions that were mentioned in chapter 2 of this document. While the low cost standalone solutions poses many limitations and the high end interconnected solutions are expensive and provides no room for after market improvements, this system we provided is cost-efficient due to its upgradability, as it allows user modifications to the system itself even beyond what this project provided brought about it being primarily open source. This system also features the common functionalities that can be found in systems management of interconnected devices thus proving its potential to lead to be part of an alternative thermal imaging platform to the solutions we have today. Some steps that may lead to achieving this is to address some points of improvement such as to deploy the system outside a local area network, to make use of jQuery on the web app for real time updates, and to integrate this system with other aspects of this project[68].

To conclude, this aspect of the project was able to achieve its higher objectives as well: to demonstrate the advantages of an interconnected, open, programmable, and extensible thermal imaging platform in terms of manageability. Open-source projects like Flask and MQTT was what primarily helped us develop this project. The programmability and interconnectivity of the nodes was what made it possible to monitor and configure them. Lastly, extensibility is what will allow the use of this management system in conjunction with the other aspects of the project.

4.2 Accuracy - Thermal imager Optimization

To determine if improving the accuracy of cheap thermal imaging systems is possible, we used an AMG8833 thermal camera for our tests. The AMG8833 has a resolution of 8x8 pixels, with a thermal range of $0-80°C$, an accuracy of $\pm 2.5°C$ and a noise equivalent temperature difference (NETD) of 0.05K-0.16K [69]. The intended use of the AMG8833 is to detect human movements or hot spots in a specified region [70]. The sensor is not designed for thermal screening purposes but has the core functionality of a thermal camera, which makes it suitable for our tests. If improving the accuracy of the AMG8833 is possible, then improving similar commercial off-the-shelf thermal cameras in the market should also be possible

4.2.1 Methodology

. The setup included a Raspberry Pi 4, coupled with the AMG8833 thermal camera and a Raspberry Pi Camera V2. We ran the setup using Raspberry Pi OS and tested the system remotely via Secure Shell Protocol (SSH). For the construction of the dataset and deep learning training, we used a powerful desktop computer running on Ubuntu with 32GB RAM, M.2 SSD Storage, AMD Ryzen 3700x processor and Nvidia RTX 2070 GPU. We used Python as the main programming language and included Numpy, OpenCV, Scipy, Tensorflow, and Keras to handle and train multiple image data.

Below is the basic overview of the accuracy methodology process for the thermal system.

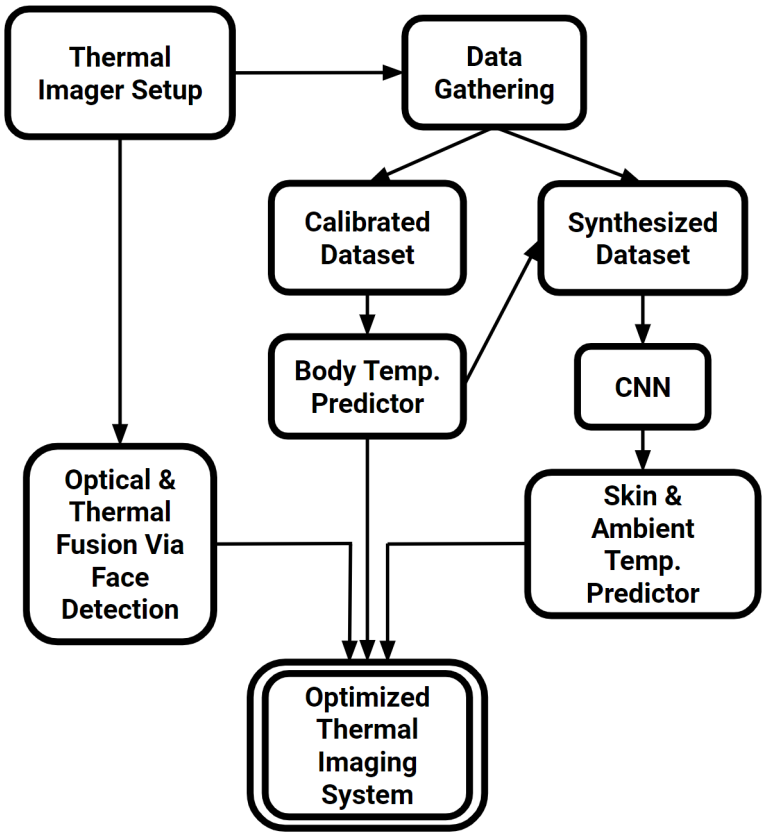


Figure 4.6: Overview of the Accuracy Methodology Process.

The first step was to setup the thermal imager, which was used to gather data. This data was then calibrated for the Body Temperature Prediction. Next, To properly train the Convolutional Neural Network (CNN), A larger dataset with different skin temperatures and multiple targets was required. This was termed the synthesized dataset. The CNN would be trained to output the skin and ambient temperature. Going back to the setup, Face detection was intergrated to fuse the Optical and thermal camera together and was used to determine the target distance and position. The final full optimized thermal imaging system is made by combining all the features together. While building this optimized thermal imaging system, we also evaluated the thermal data, compared different CNN techniques, and determined if having multiple targets affected the accuracy.

To get a better picture of the thermal system, below is the final thermal imager model.

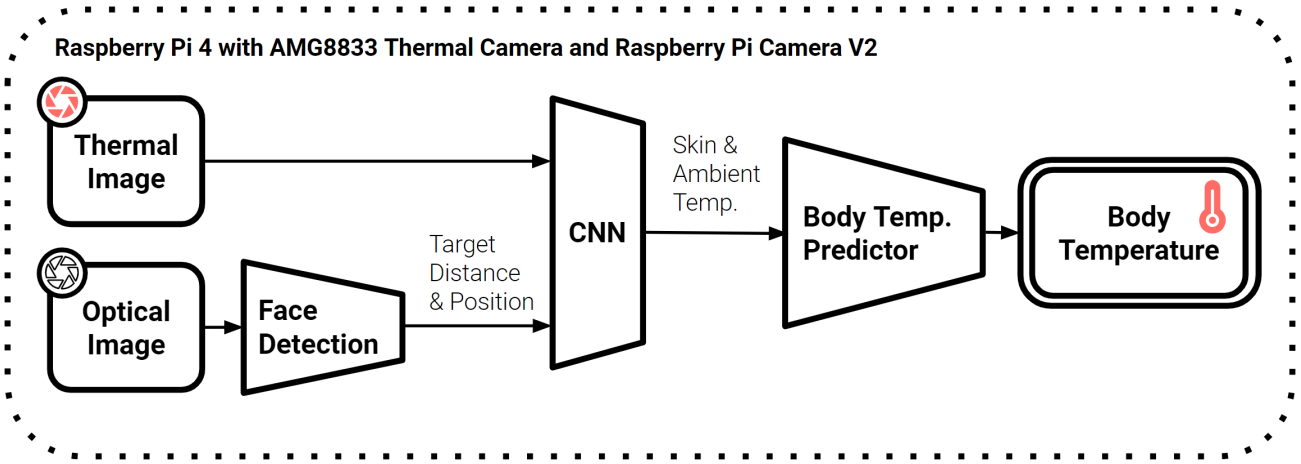


Figure 4.7: Thermal Imaging System Model

The Raspberry Pi streams optical and thermal images using the Rpi Camera and the AMG8833 respectively. The optical image is fed into a face detection algorithm, which outputs the target distance and position to the CNN, along with the thermal image. The CNN then outputs the skin and ambient temperature to the body temperature predictor, which gives the final body temperature estimation.

4.2.1.1 Gathering Data

There were three kinds of tests made using the AMG8833: the ice bath test, the 8-hour idle test, and the ambient temperature - distance test.

The ice bath test was conducted by placing a cup full of crushed ice submerged in cold water under the thermal sensor. Theoretically, the ice bath should be $0°C$. Multiple tests of 100 frames taken in 10 seconds was conducted and averaged in time. This average is the sensor error.

The 8-hour idle test was conducted by putting a one cubic foot cardboard box around the sensor to prevent any wind from affecting the readings. The box was also used to make the temperature readings uniform. The sensor was left overnight for eight hours, recording at ten frames per second. This test was to determine the temporal noise present and the fluctuation in temperature over time.

The ambient temperature - distance test was made to determine the relationship among ambient temperature, target distance, target position and the temperature readings. These tests were used as the raw dataset for the statistical and machine-learning processes. Each test had 100 frames taken within 10 seconds. All frames within each test had a consistent target position and a consistent ambient temperature. Four sets of varying ambient temperatures were conducted. Each set had four subsets of varying target distances. These distances were determined by the pixel area of the head as viewed from the camera. The four distances were 1x1, 2x2, 3x3, and 4x4. All 1x1 subsets had a combination of six horizontal and six vertical positions (total of 36). For every increment in head size, the number of horizontal and vertical positions reduced by one. This gave a total of 25, 16, and 9 position combinations for 2x2, 3x3, and 4x4 respectively. To make the dataset more evenly distributed, the 2x2, 3x3, and 4x4 subsets were linearly interpolated to have the same number of positions combination as the 1x1 subset. This gives a total of

57,600 total thermal images. To evaluate the accuracy later on, the core body temperature of the target was obtained using an oral thermometer. Additionally, all four varying temperature sets contained a test with no target.These were used later on for calibrating the dataset.

Note that these tests were very prone to human error. We assumed that the test taken at the same time period all had the same body temperature and same ambient temperatures. We also assumed that the positions and distances taken were all precise. Of course, these most likely were not the case. By taking the averages, we hoped to reduce these errors.

4.2.1.2 Body Temperature Prediction Using Statistical Approaches

The next step was to make a function that calculates the body temperature using the ambient temperature, skin temperature and target distance. The raw data gathered has a known target distance. However, the ambient temperature and skin temperature are unknown. Using only the raw thermal output to obtain the ambient temperature and skin temperature is ineffective as it contains a lot of noise.

We used this simple equation below to compute for the denoised temperature reading E_{sensor}. Note that *T* is temperature, *E* is error, and *N* is noise.

$$T_{denoised} = T_{raw} - E_{sensor} - N_{sensor} - N_{wind} \tag{4.1}$$

Sensor error E_{sensor} was obtained by getting the mean temperature of the ice bath test over time as seen in the equation below.

$$E_{sensor} = mean(T_{ice}) = \frac{\sum_{i=1}^{n} T_i}{n} \tag{4.2}$$

Unlike E_{sensor}, sensor noise N_{sensor} and wind noise N_{wind} were unobtainable. Fortunately, getting the mean and maximum of the raw temperature T_{raw} will remove N_{sensor} and N_{wind} respectively. This is because N_{sensor} is centered at zero while N_{wind} is always negative when present and zero when not present. Given that E_{sensor} was removed and $T_{denoised}$ is constant, the following equations below demonstrate how the noise is omitted. Note that these equations only work if the temperature data recorded spans over a substantially long period in time.

$$mean(T_{raw}) = T_{denoised} + mean(N_{wind}) \tag{4.3}$$

$$max(T_{raw}) = T_{denoised} + max(N_{sensor}) \tag{4.4}$$

In the 8-hour idle test, we made sure that there was no wind present. Therefore, using Eq. 4.3, $mean(T_{raw}) = T_{denoised}$. Substitute this value to $T_{denoised}$ in Eq. 4.4 and we can get $max(N_{sensor})$. With this, we can compute the value of $T_{denoised}$ given any large set of raw temperature data. Combining all the equations, we can compute for E_{sensor} as seen in the equation below.

$$T_{denoised} = max(T_{raw}) - max(T_{idle}) + mean(T_{idle}) - mean(T_{ice}) \tag{4.5}$$

Using Eq. 4.5, the ambient temperature - distance dataset is denoised by using the 100 frames per test to come up with one denoised thermal image per set. This reduced the number thermal images from 57600 to just 576. However, unwanted wind noise could still be present throughout all the 100 frames. Eq. 4.5 is only effective in reducing the wind noise up to the minimum wind noise present in any of the 100 frames. Because of this, some tests were more denoised than others. Despite there being only 100 frames per test, there were $(36 * 25 * 16 * 9) * 100 = 8600$ frames per ambient temperature set. We took advantage of this since the chances that there was no wind noise in at least one of the 8600 frames was very likely.

We used Gaussian Gradient Descent (GMM)[71] to level all the tests to match the maxed denoised test. First, we subtracted the target-less sample from the thermal image. Next, we increased the resolution through bicubic interpolation for more pixel data, rounded all values to the nearest quarter (0.25), then got the histogram distribution. Using GMM, we clustered the histogram into two Gaussian curves: the lower curve being the ambient distribution, and the upper curve being the target distribution. The x-value where the ambient distribution peaked is the average difference in ambient temperature between the thermal image and the target-less sample. Lastly, we leveled all the test by subtracting its thermal image to the peak ambient distribution x-value then adding it to the highest x-value among the tests within the same ambient temperature. This levelling process can be explained using the equation below where *W* is the weight of the ambient distribution.

$$test_i += max(argmax(W_0), argmax(W_1), ..., argmax(W_n)) - argmax(W_i) \tag{4.6}$$

Instead of using an external thermal reading to get the ambient temperature, we used the thermal image data to determine the ambient temperature. Utilizing the ambient temperature GMM distribution curve mentioned in the last paragraph, we took the y-value as the weights. These weights tell how close the pixel readings were to the ambient temperature. We took the weighted average to get the ambient temperature as seen in the equation below.

$$T_{ambient} = \frac{\sum_{i=1}^{n} T_i * W_i}{\sum_{i=1}^{n} W_i} \tag{4.7}$$

We also calculated the ambient temperature error, which is the difference in the maximum and minimum calculated ambient temperature. The true ambient temperature reading was set to the largest calculated ambient temperature among the tests that were taken with the same ambient temperature.

The skin temperature was obtained by increasing the thermal image thermal resolution with bicubic interpolation, then taking the maximum temperature within the target area. This target area was determined using GMM once again. Note that this is not actual skin temperature as the thermal readings alter with different ambient temperatures and target distances.

Using the ambient and skin temperature obtained from the denoised thermal images, together with the target distance and position, we plotted the ambient temperature vs skin temperature, distance vs skin temperature, and position vs skin temperature. As there are only a few ambient temperature and distances tested, bicubic interpolation was used to bridge the gaps in between. These plots were used to measure the accuracy of the skin temperature readings. We repeated the plotting process but instead of skin temperature,

we compared the data to the skin temperature offset. The offset was used to determine the precision of the skin temperature readings. The offset was computed using the equation below.

$$\text{Skin Temperature Offset} = (max(T_{skin}) - min(T_{skin})) * 0.5 \tag{4.8}$$

Aside from evaluating the accuracy and precision of the thermal camera, the plots were used to fit a model to predict the core body temperature, given that the target distance, ambient temperature and skin temperature were known. This process was done by fitting a third degree polynomial using the four target distances tested and the difference between body temperature and skin temperature calculated. This process was repeated four times with the four ambient temperatures tested. The next step was to plug in the input target distance to the polynomials. The answers to the four polynomials were then used to fit another polynomial. We then plugged in the input ambient temperature to the polynomial and added the answer to input skin temperature. This would give the predicted body temperature. This process is essentially a bicubic interpolation.

4.2.1.3 Skin and Ambient Temperature Prediction Using Statistical Approaches

Before tackling this problem using CNN, we first tried a full statistical approach. There are countless of ways to predict the skin and ambient temperature using statistical approaches. Some are more accurate than others, some perform faster. This project used the minimum-maximum approach which converts the minimum and maximum temperatures of the thermal image to ambient and skin temperatures respectively. This approach was chosen as it is very easy to calculate. The purpose of making a full statistical approach is to have a comparison in accuracy performance for the later CNN approaches. The CNN approach should have better results or else there would be no point in making one.

The way the minimum-maximum statistical approach was implemented was first take the minimum and maximum of all thermal images in the raw dataset. The minimum was plotted along with the set ambient temperature and the maximum was plotted along with the set skin temperature. The set ambient and skin temperatures were calculated from the calibrated dataset made in the previous section. A line was then fitted on the two plots which served as the mapping function from minimum-maximum temperatures to ambient-skin temperatures.

To evaluate the accuracy of this method, the error of the skin and ambient temperature outputs were calculated by taking the difference between its output temperatures and set temperatures. These errors were then placed in a histogram chart to observe the shape, mean, and variance of the error. the Body temperature error was also evaluated, which was measured by taking the skin and ambient temperature outputs and plug it to the body temperature predictor made in the previous section.

One of the limitations of this statistical method is that it does not it cannot take multiple target estimations at the same time.

In the next section on CNN, the same evaluation process was used to compare the accuracy performance.

4.2.1.4 Skin and Ambient Temperature Prediction Using CNN

In the previous section, body temperature is predicted using the target distance, ambient temperature, and skin temperature as the input. This section covers how to get ambient temperature and skin temperature given a raw thermal image. There were two approaches tested: one was supervised, the other was unsupervised. The supervised learning approach used the target distance and position as the labels to help find the skin and ambient temperature. The unsupervised learning approach only had the raw thermal image as the input and not only did it output the skin and ambient temperature, it also output the target distance and position. Unsupervised learning allows the model to work on its own to discover patterns and information that supervised learning would not detect [72]

For both approaches, Convolutional Neural Networks or CNNs was used. CNNs are generally used for handling images as they are good in capturing spatial dependencies [73]. This similar but different from the standard feed forward network or multilayer perceptrons (MLPs). MLPs can also work on images but not as good as CNNs in general.

Below is a basic diagram that differentiates the two kinds of CNN models:

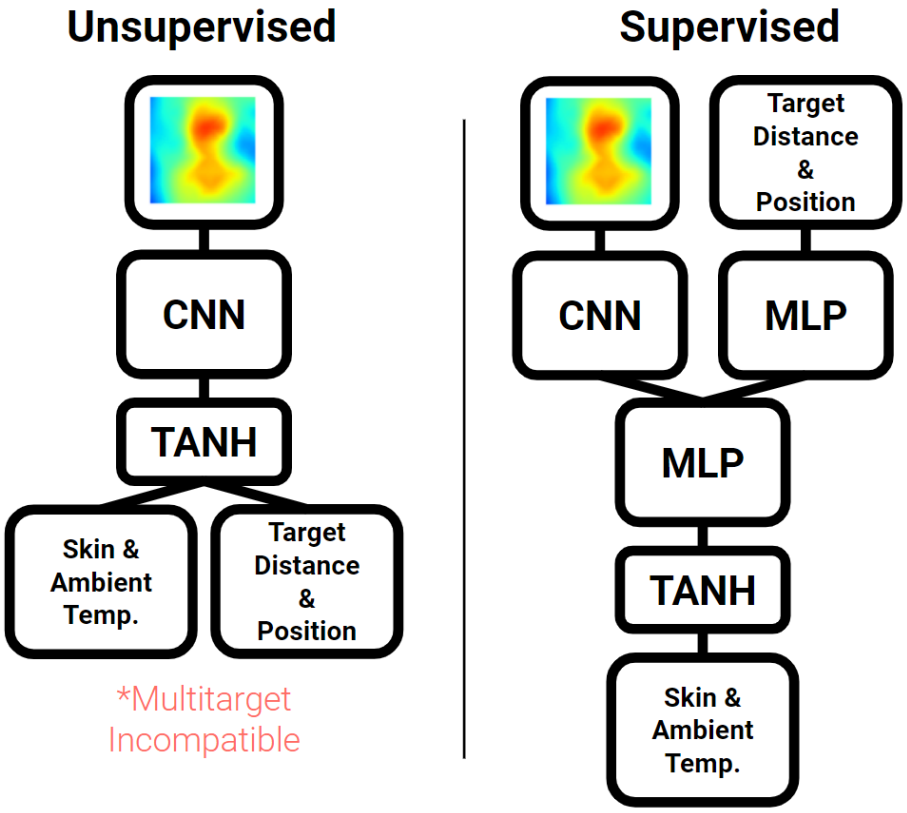


Figure 4.8: Unsupervised and Supervised CNN Models

As mentioned in the diagram above, the unsupervised learning approach is not compatible with multiple targets. This is because if there are multiple targets on one thermal image, the CNN model would not know which target it is supposed to measure. In the supervised learning, the label not only acts as a

guide to help train the CNN model, but also provide the target details so the model would know which target it is measuring.

In the supervised learning approach, some parts of the model uses the MLP instead of CNN. This is simply because The data handled by the MLP are not images anymore. The CNN convolves the thermal image several times until the image is reduced to a list of single pixels. This list of pixels is then merged with the target labels and goes through a short set of MLP.

Both models go through a final Tanh activation layer. The Tanh function has an output range of (-1,1). Tanh and Sigmoid are the usual final activation layers when training for regression model training like this one. Regression models are used when dealing with continuous data like temperature. This is different from classification models that are more binary in nature. Using Sigmoid as the final activation layer would also be valid, but based on studies, Tanh usually outperforms sigmoid [74]. Since the predicted temperatures go beyond the Tanh output range or (-1.1), the output of the model is mapped to larger range. In this case, the skin temperature was mapped to (22,36) while ambient temeprature is mapped to (0,50).

For the optimization, both models used ADAM with a learning rate set to 3e-4. As these are regression models, mean absolute error (MAE) was used as the loss function. Mean squared error (MSE) was also tested as the loss function, but MAE performed slightly better.

The equations for MAE and MSE are as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2$$

Where,

$\hat{y}$ $-$ *predicted value of* y

$\bar{y}$ $-$ *mean value of* y

Multiple CNN model structures were tried and tested. The unsupervised and supervised models that performed best are shown below

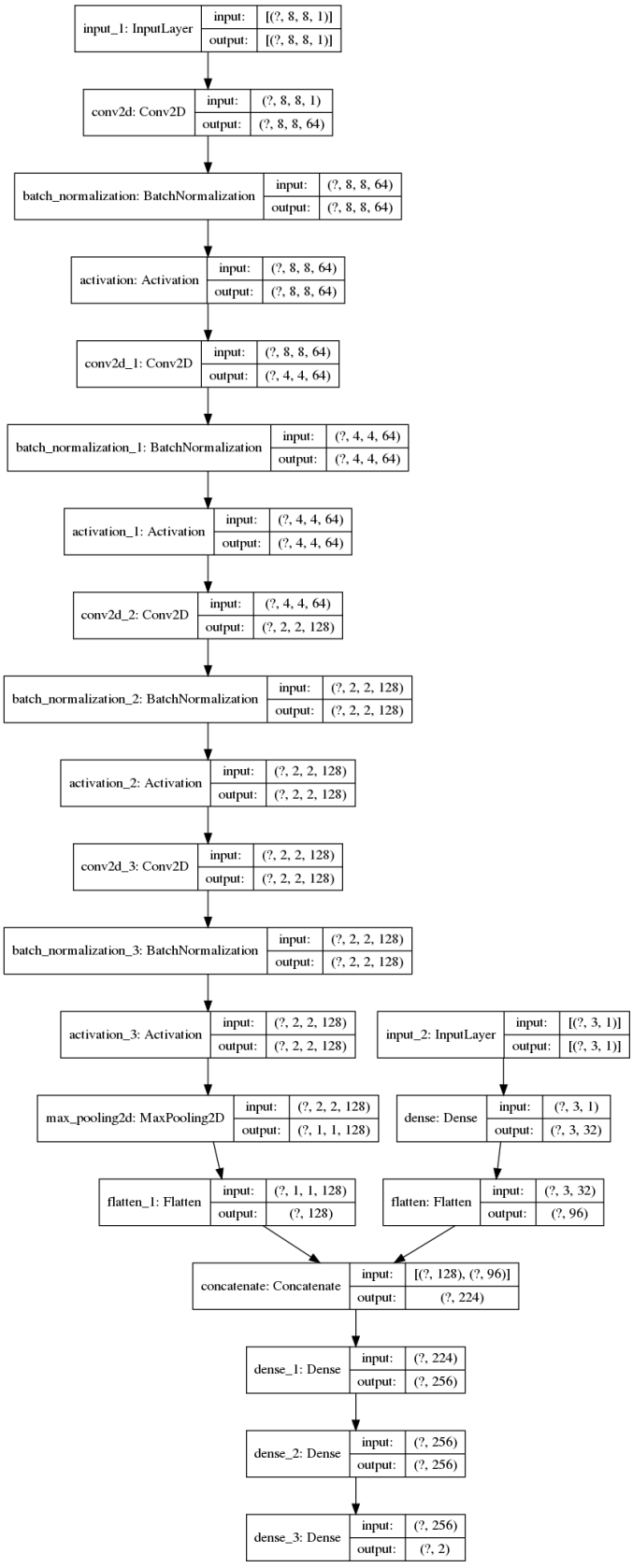


Figure 4.9: Full Supervised CNN Model

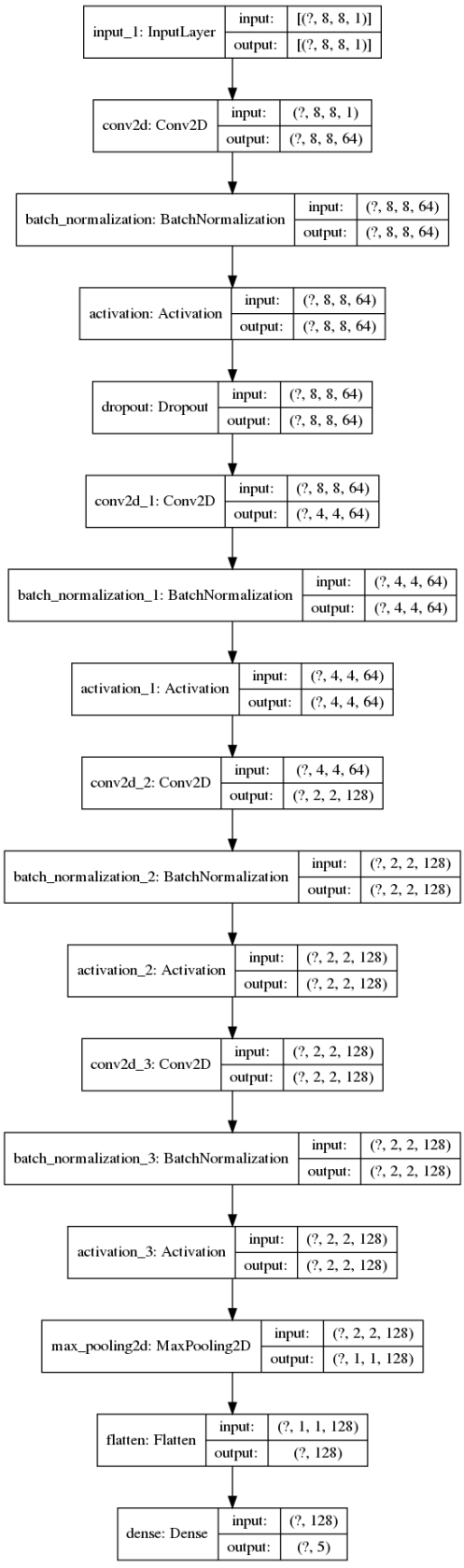


Figure 4.10: Full Unsupervised CNN Model

The CNN part of both models are the same except the Unsupervised has a dropout layer of 0.5 near the top of the neural network. The CNN is structured with four convolution sets, ending with max pooling. A convolution set is structured is structured as follows:

$$Convolution- > BatchNormalization- > Relu \tag{4.9}$$

All convolutions except the first have strides of two.

The raw dataset was not enough to train the CNN as it was taken with a single target at about $36.5°C$. The CNN needed to be trained on varying body temperatures or else it would not know how to detect fever. It also needed to be trained on thermal images with multiple targets to make the CNN predict multiple target temperatures simultaneously. The CNN also would benefit more if there were not only four ambient temperatures and four distance values taken during the data gathering.

Due to the limitations in facilities, equipment, and human volunteers of varying body temperatures, a lot of thermal images were synthesized to satisfy the CNN training requirements. To solve the limited ambient temperature and target distance data gathered, thermal images were linearly interpolated to produce more variety of of thermal images. The number of ambient temperatures and target distances was both bumped from 4 to 16.

For multiple targets, two thermal images taken in the same ambient temperature were merged together by taking the hotter pixel between the two thermal images. Ideally, the two images would be levelled using the GMM method earlier before merging, but the process of levelling every possible pair combination would take a lot of time. To prevent overlapping targets, a boundary box was calculated based on the target distance and position. Two targets are considered overlapping if their boundary boxes intersect.

The problem with synthesizing new thermal images with new body temperature was that there was only one body temperature tested. If there was at least two body temperatures, synthesizing the in-between body temperatures through linear interpolation would have been possible. To solve this problem, temperatures near the ambient temperature remained the same while those closer to the skin temperature were adjusted. Again, using GMM to classify which pixels are background and which pixels are target would be more accurate, but would take a lot of time to process the large synthesized dataset. The problem with both methods, or any body temperature synthesizing technique with only one body temperature measured is that the rate of change of skin temperature reading over body temperature is unknown and was estimated. With that said, training the CNN model with this synthesized data will most likely not give the correct result the further it goes away from $36.5°C$. However, for the sake of testing and evaluating the CNN, we assume that the synthesized thermal data is correct.

For training 80% of the synthesized dataset was used for training, 10% on validation and 10% on test. The dataset was randomized so the three sets include a good mixed variety of thermal images. The training composed of 200 Epochs with a batch size of 256.

There was a total of 10,598,400 thermal images in the synthesized dataset. Comparing it to the 57,600 thermal images in the raw dataset, only 0.54% was not synthesized. In fact, the total number of possible merged thermal images without even varying the body temperature is about 26.5 billion. This would definitely exceed the memory limit of most computers. Even 10 million thermal images was a lot

that having a 16GB RAM was insufficient. Fortunately, training was done with a 32GB RAM which made training 10 million thermal images possible.

Since the unsupervised learning model is not suitable for multiple target estimations, a different synthesized dataset was used. This one had 8,294,400 total thermal images. Nine different body temperatures from $35°C$ to $40°C$ were synthesized. Unlike the multiple target dataset, the single target dataset was able to fit all possible combinations in memory.

4.2.1.5 Optical and Thermal Fusion and Integration

To further improve the accuracy of our thermal imaging system, we fused the optical and thermal cameras so we can harness high-resolution capabilities of the optical camera such as face detection. But before we can utilize fusion of cameras, we needed to properly map the optical images to the thermal images. This was done by simply putting the two cameras as close as possible on the same vertical position. The lenses are about an inch apart. Fortunately, both cameras have nearly the same horizontal field of view (FOV), with the amg8833 having a $60°$ FOV and the RPi Camera V2 having $62.2°$[69] [75]. Unfortunately, the vertical FOVs were different, which meant that a top and bottom part of the thermal images were not be covered by the optical camera.

We used the Face Recognition Library provided by Adam Geitgey, built using dlib's state-of-the-art face recognition [76][77]. We only used the library for face detection, required for real-time face location and distance measurement. The face detection algorithm can also detect multiple targets simultaneously, which will allowed multiple target estimation.

Based on figure 4.7, The CNN model and the body temperature predictor were integrated together in the Raspberry Pi. After integration, the performance of the thermal system was evaluated by scanning targets at different distances and ambient temperature. The thermal system was also tested if multiple target estimation works and if it could detect fever. Since there was no sick person to test, the raw thermal image was modified similar to how the synthesized data modified skin temperatures of the raw dataset.

4.2.2 Preliminary Results

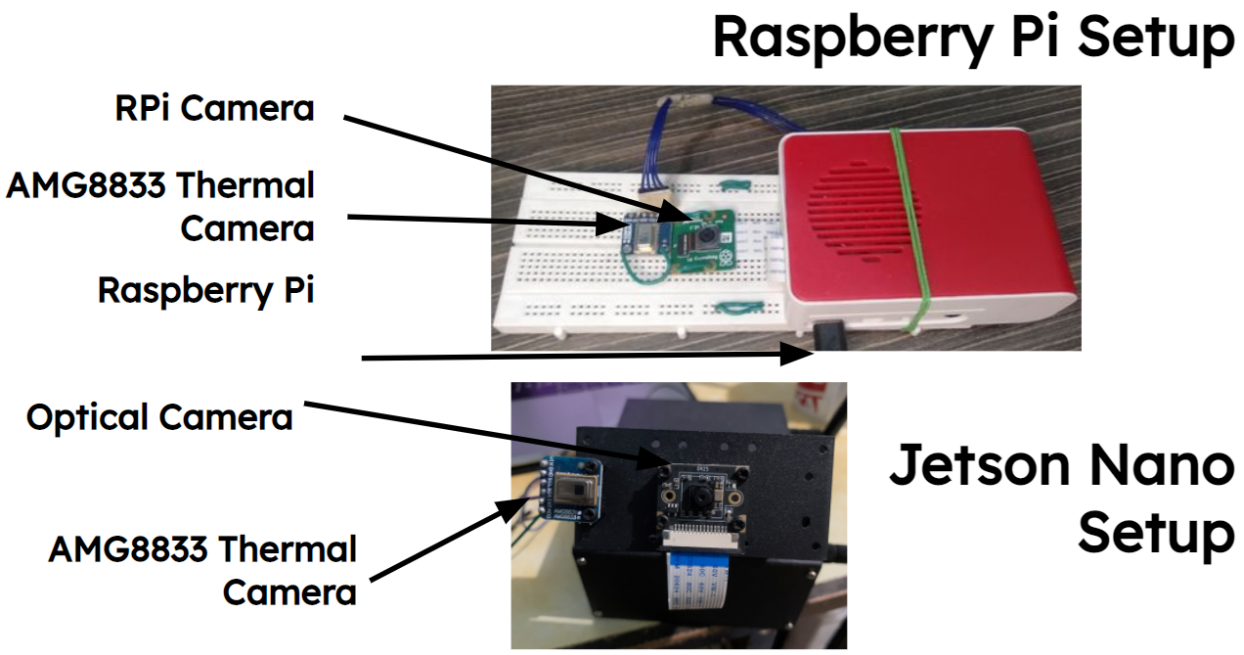


Figure 4.11: Preliminary Setup for Raspberry Pi and Jetson Nano

To get a better picture of the tasks at hand and the problems that we would encounter, we built two setups for remote thermal screening; one uses a Raspberry Pi, while the other uses a Jetson Nano. Both setups have an optical camera and an AMG8833 thermal camera installed.

4.2.2.1 Preliminary Features

We built our thermal imaging system to have the following features for initial testing:

- Merged optical and thermal output
- Bicubic interpolation of thermal output
- 8x8 thermal readings display
- SSH connection for remote control

Face detection was later integrated into the setup after initial testing.

4.2.2.2 Installing the AMG8833 Thermal Camera

Our first concern was how to read the AMG8833 outputs. The thermal camera communicates with the mini-computers via I2C. We properly connected the input/output (I/O) of the AMG8833 to the General-

purpose input/output (GPIO) pins of the two mini-computers. Though the orientation of the GPIO pins differ between the two mini-computers, both have the necessary pins for proper connection.

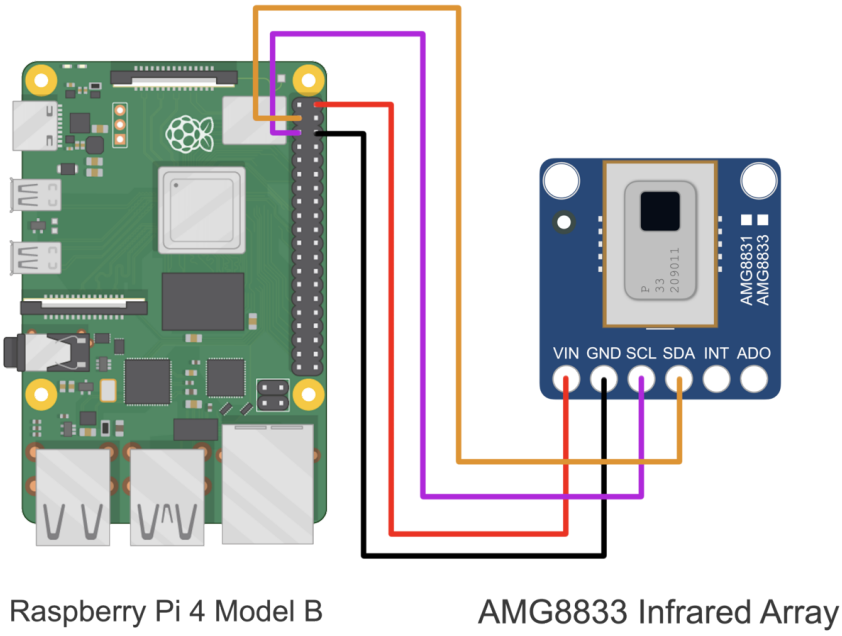


Figure 4.12: Wiring guide for Raspberry Pi.[78] Jetson Nano wiring is similar to this.

After we detected connection between the AMG8833 and the mini-computers, we installed the AMG8833 python library provided by Adafruit [79]. This library provided us with the essential functions to initialize the AMG8833 and the function to read the 8x8 thermal readings using Python code. The library is based from the AMG88xx reference sheet provided by Panasonic [80]. We also installed other libraries such as OpenCV, scipy and imutils to make thermal image handling easier.

4.2.2.3 Initial Testing

For ease of access, we ran the tests remotely via SSH using MobaXterm. The first tests were to see if the AMG8833 works. We checked whether the sensor can detect hot spots and humans at room temperature. We then tested if the thermal sensor can detect extremely hot objects like a soldering iron and freezing temperatures of ice cubes.

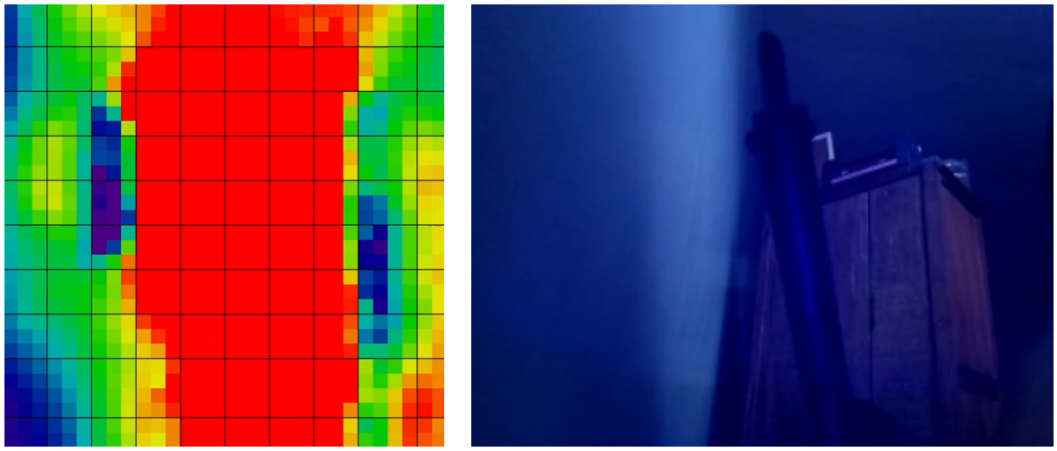
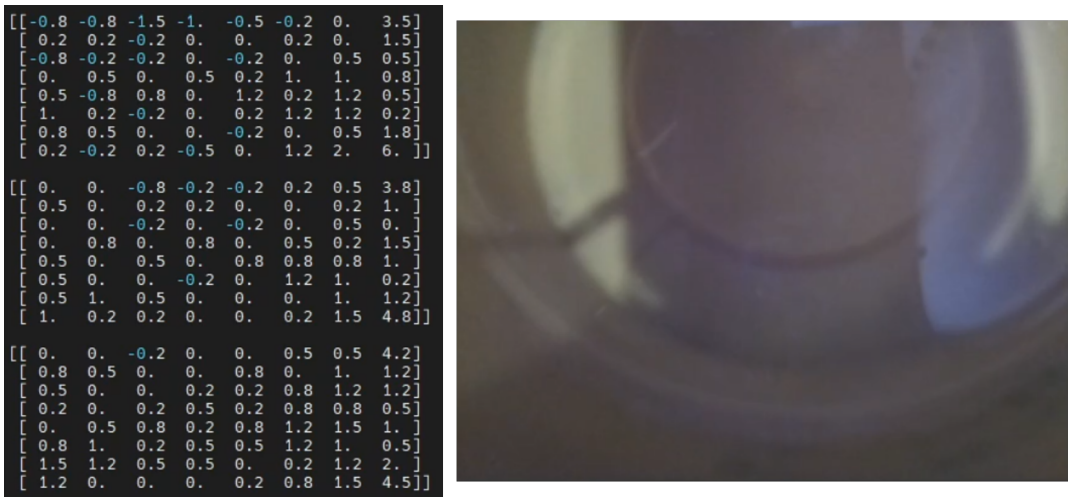


Figure 4.13: thermal camera sensing soldering iron.

While we were taking the temperature of ice cubes, we proceeded to do the ice bath test. When ice melts, the temperature of the surrounding water is theoretically at $0°C$ and will remain at $0°C$ until all the ice melts. We followed the instructions given by ThermoWorks [81] to create our own ice bath. However, we had limited resources and did not access to crushed ice as suggested so we froze water in a bowl and made it slowly melt. Another problem was that the ice was floating on the water, which meant that the thermal camera was scanning the ice and not the freezing water. Since we cannot make an ice well as suggested by ThermoWorks, we lifted the large melting ice out of the water then immediately scanned the water. Our readings were roughly $0°C$ but the temperature had a tendency to fluctuate.



```
[[-0.8 -0.8 -1.5 -1.  -0.5 -0.2  0.   3.5]
 [ 0.2  0.2 -0.2  0.   0.   0.2  0.   1.5]
 [-0.8 -0.2 -0.2  0.  -0.2  0.   0.5  0.5]
 [ 0.   0.5  0.   0.5  0.2  1.   1.   0.8]
 [ 0.5 -0.8  0.8  0.   1.2  0.2  1.2  0.5]
 [ 1.   0.2 -0.2  0.   0.2  1.2  1.2  0.2]
 [ 0.8  0.5  0.   0.  -0.2  0.   0.5  1.8]
 [ 0.2 -0.2  0.2 -0.5  0.   1.2  2.   6. ]]

[[ 0.   0.  -0.8 -0.2 -0.2  0.2  0.5  3.8]
 [ 0.5  0.   0.2  0.2  0.   0.   0.2  1. ]
 [ 0.   0.  -0.2  0.  -0.2  0.   0.5  0. ]
 [ 0.   0.8  0.   0.8  0.   0.5  0.2  1.5]
 [ 0.5  0.   0.5  0.   0.8  0.8  0.8  1. ]
 [ 0.5  0.   0.  -0.2  0.   1.2  1.   0.2]
 [ 0.5  1.   0.5  0.   0.   0.   1.   1.2]
 [ 1.   0.2  0.2  0.   0.   0.2  1.5  4.8]]

[[ 0.   0.  -0.2  0.   0.   0.5  0.5  4.2]
 [ 0.8  0.5  0.   0.   0.8  0.   1.   1.2]
 [ 0.5  0.   0.   0.2  0.2  0.8  1.2  1.2]
 [ 0.2  0.   0.2  0.5  0.2  0.8  0.8  0.5]
 [ 0.   0.5  0.8  0.2  0.8  1.2  1.5  1. ]
 [ 0.8  1.   0.2  0.5  0.5  1.2  1.   0.5]
 [ 1.5  1.2  0.5  0.5  0.   0.2  1.2  2. ]
 [ 1.2  0.   0.   0.   0.2  0.8  1.5  4.5]]
```

Figure 4.14: Ice bath test

Seen in the image above, the 3 sets of 8x8 array represents the temperature readings for the last 3 frames. Temperature fluctuation was evident.

Next, we took note of how far a person is detected before they blend into the ambient temperature.

Based on our tests, when a person stands about 1.3 meters away from the camera, we could hardly detect the face temperature. We believe this is caused by the target face being smaller than a single sensor region.

Figure 4.15: Thermal camera barely sensing target face.

During a pandemic, people wear masks and face shields. We wanted to see whether the thermal camera can detect face temperatures with face masks and face shields. We also wanted to test whether face detection works with occlusion. We used Adam Geitgey's Face Recognition library for these tests.

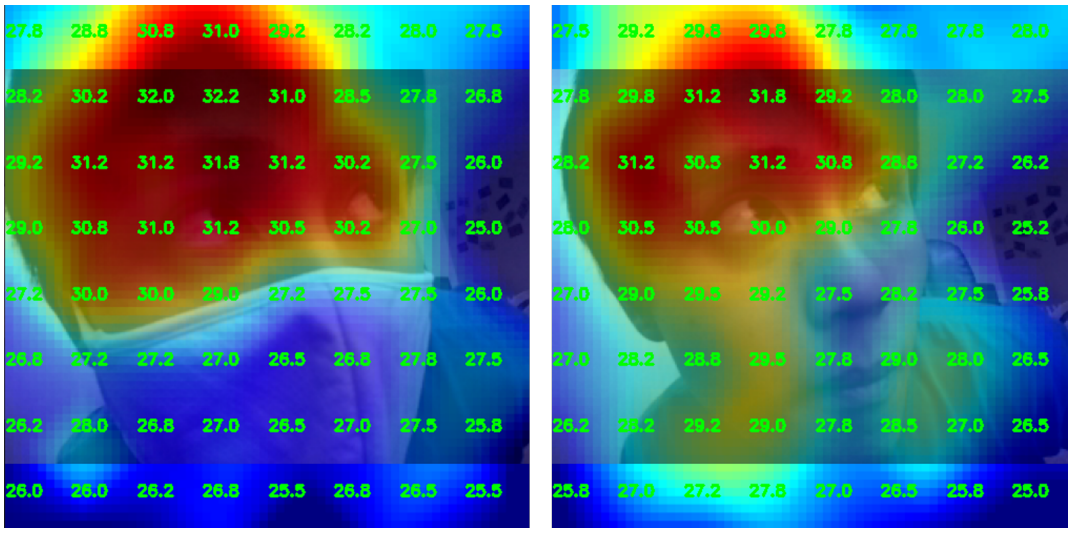


Figure 4.16: Face Mask VS No Face Mask

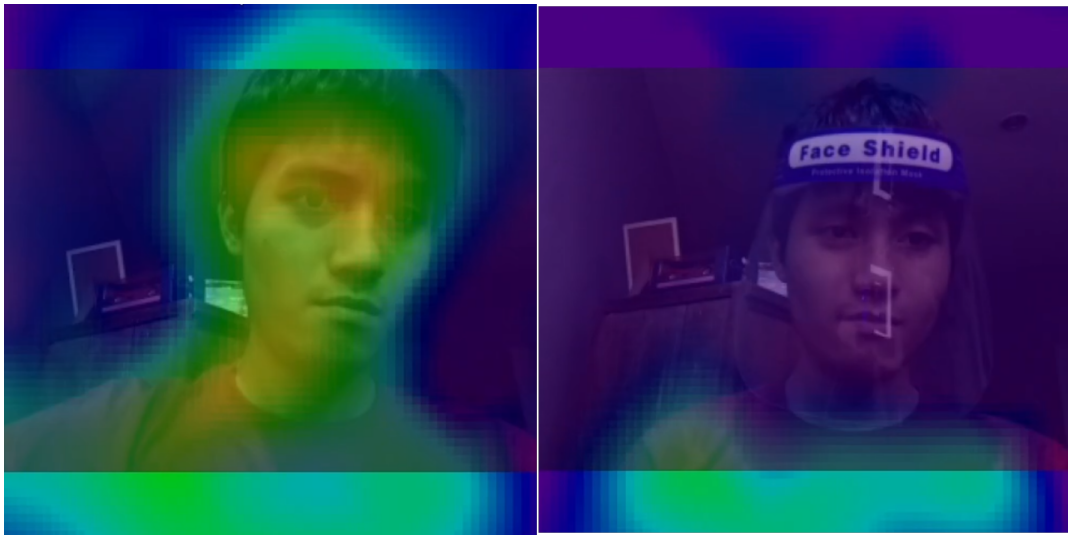


Figure 4.17: Face Shield VS No Face Shield

From the figures, we see that face mask and face shields do block temperature readings. Fortunately, face masks only block the lower part of the face which is usually cooler than the upper face. Face shields unfortunately make reading face temperatures impossible.

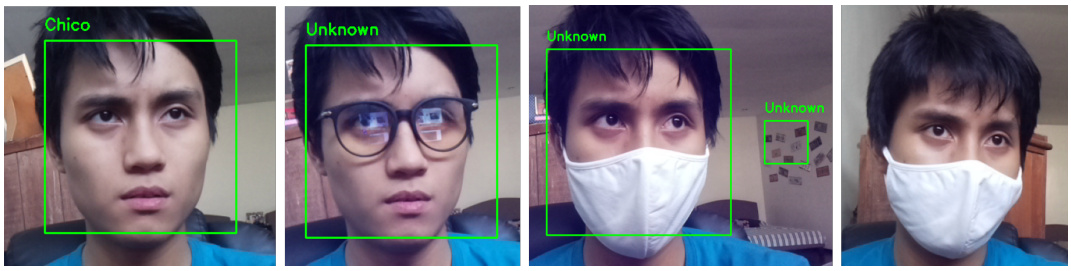


Figure 4.18: Face Recognition and Face Detection With and Without Occlusion

Based on the figures above, we can see that face recognition only works on non-occluded faces. Face detection is able to work on faces with glasses, but For those wearing masks, face detection may sometimes fail (as seen on right most image). Lastly, detailed backgrounds may sometimes be detected as faces.

4.2.2.4 Preliminary Findings

Here is a summary of all our findings when using a thermal camera like the AMG8833 and Adam Geitgey's Face Recognition Library. The list also includes some information not mentioned earlier.

- Skin temperatures are a lot lower than core body temperature.
- The further the target, the lower the temperature readings.
- Temperature fluctuates.
- The low resolution makes it hard to catch tiny detail.
- Face shields block the thermal readings on the face entirely.
- Hot environments makes it easier for face to blend with environment.
- Touching the AMG8833 IC affects readings greatly.
- Wind from electric-fans block thermal readings.
- Face recognition does not work on occluded faces.
- Face detection may fail with face masks.
- face detection may detect a face where there is none.
- Hand temperatures are inconsistent compared to face temperatures.

4.2.3 Thermal Data Analysis

In this subsection, we analyze and discuss the experimentation and results during the construction of the thermal imaging system.

4.2.3.1 Wind Noise Analysis

Prior to the 8-Hour idle test, we first did an open 1-Hour Test. The sensor was simply pointing at a blank background placed indoors. At this time, we were not aware how even the weakest winds can affects the thermal readings.

The figure below shows the temperature readings of 5 out of the 64 sensors of the 8x8 thermal camera taken in one hour.

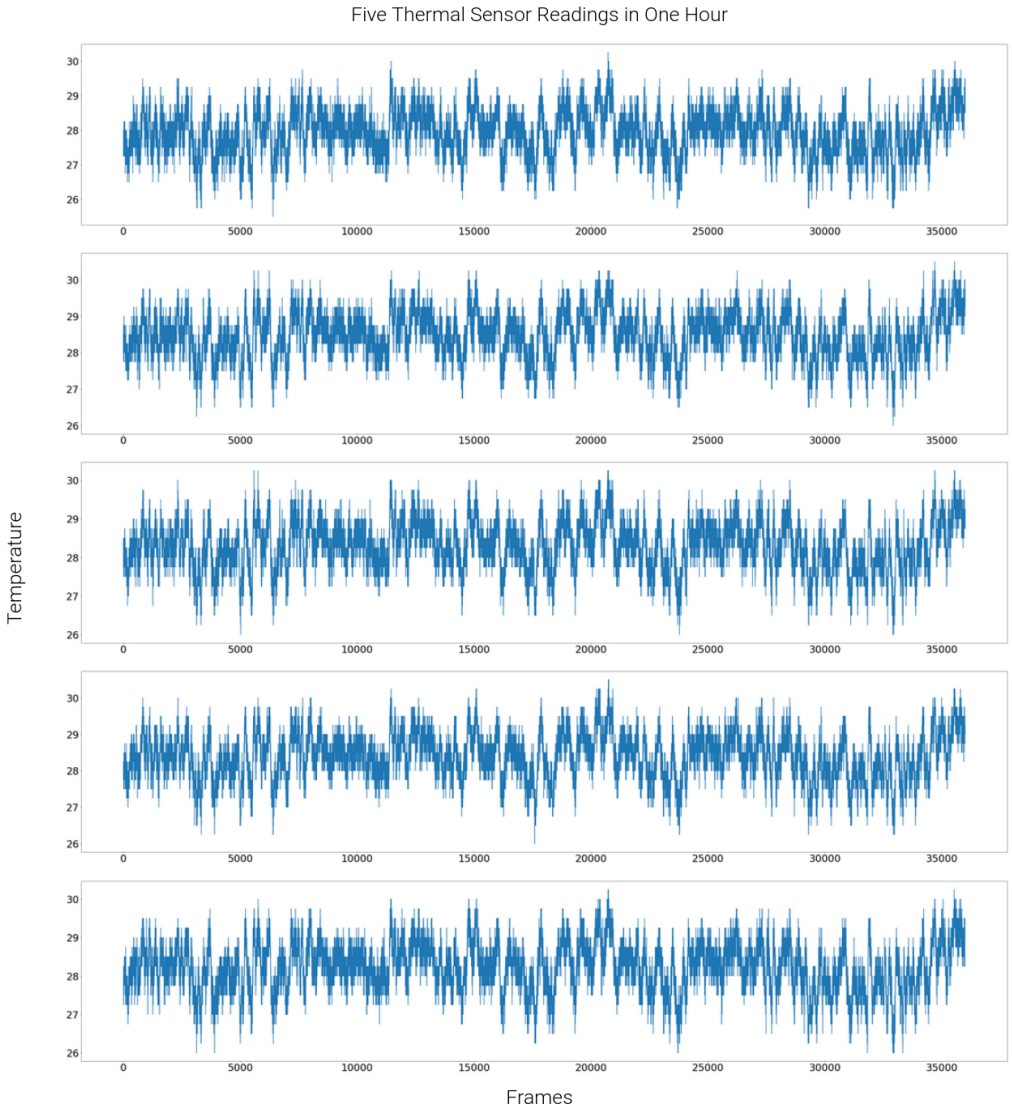


Figure 4.19: Five Thermal Sensor Readings in One Hour

Although only five sensor readings are shown in the figure, all the other 61 sensor readings follow the same similar trend. All readings look nearly identical to each other and suggests that there is a noise present in all thermal sensors. We did not know what caused this in the beginning until we did the 8-hour

idle test to confirm that it was indeed wind noise.

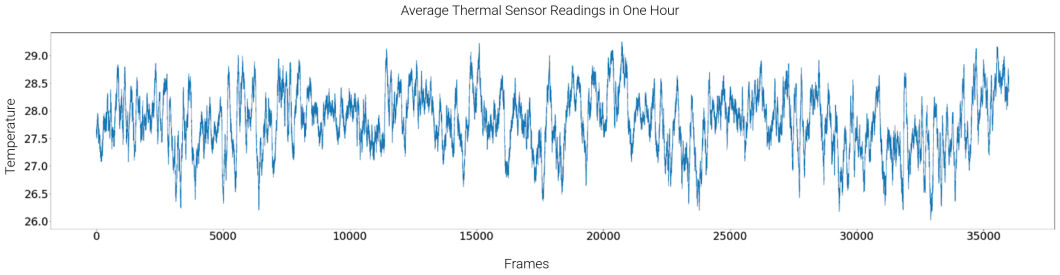


Figure 4.20: Average Thermal Sensor Readings in One Hour

The figure above shows the average readings of all the 64 senors. Notice how it was able to capture the wind noise. Averaging reduced the sensor noise to nearly zero, but the wind noise does not average to zero.

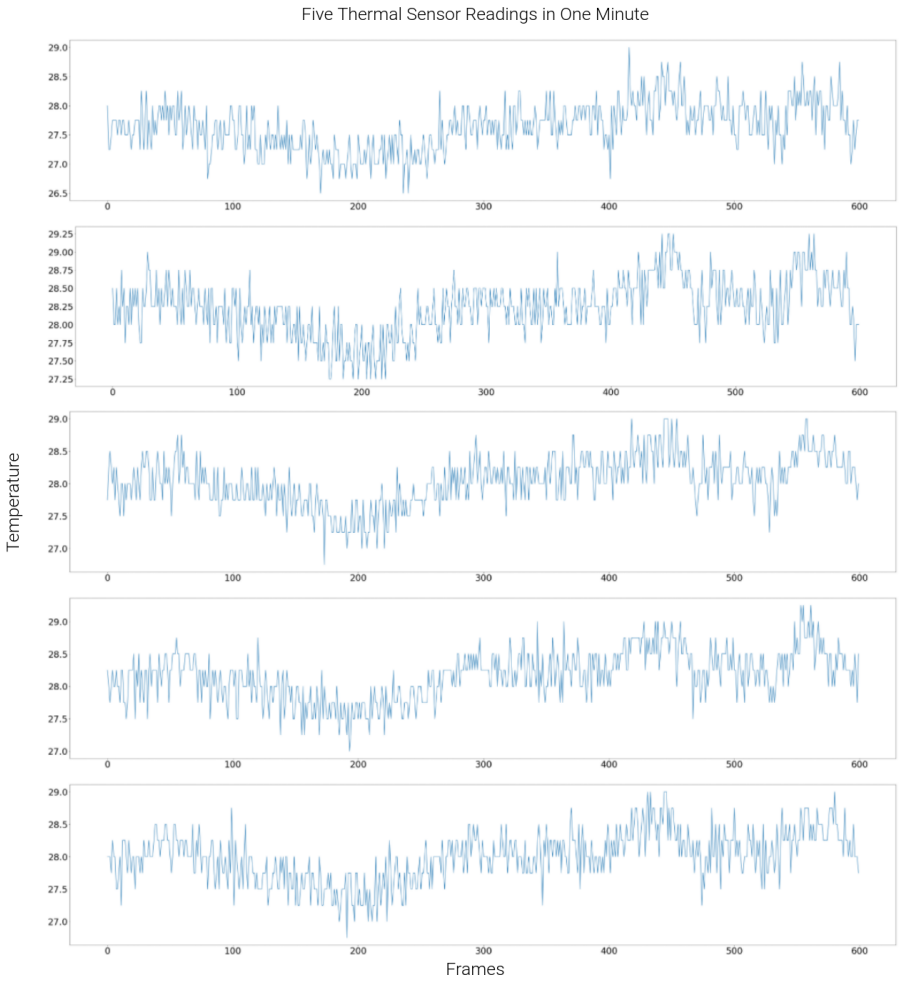


Figure 4.21: Five Thermal Sensor Readings in One Minute

We tried to see if the wind noise is still evident at a shorter time period. Instead of examining the full hour, we examined a minute time period as seen the figure above. Even at this scale, the wind noise was still present and was seen in all sensors.

The 8-hour idle test was conducted to see if there was wind noise. Recall that the 8-hour idle test has a cardboard box over the thermal camera so it would not pick up any wind. The test was conducted from 2am to 10am. Below shows our results over 3 sensors. The temperature readings in the figure are relative to the average temperature that time.

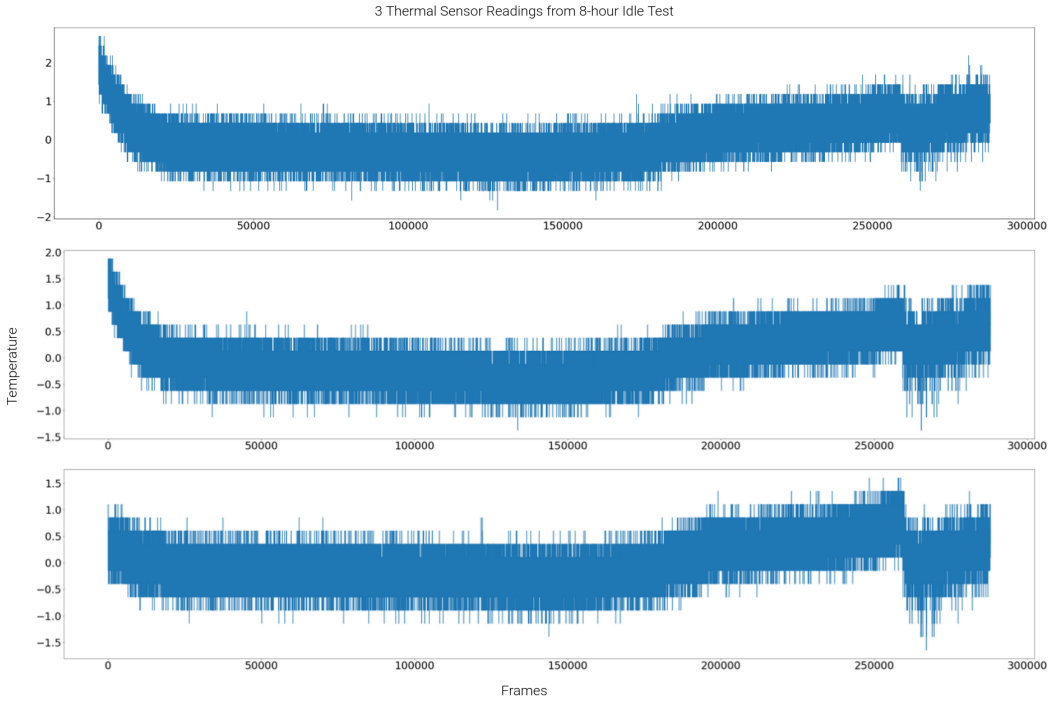


Figure 4.22: Thermal Sensor Readings From 8-Hour Idle Test

Just like the 1 hour test, the 8-hour idle test showed similar results among its 64 sensors. One anomaly spotted was how the temperatures dropped by $3°C$ in the beginning. This took around 20,000 frames, which is equivalent to about half and hours time. This was obviously not the case, especially in a country like the Philippines where the test was conducted. Possible causes could be the air inside the box cooling down at a faster rate than the outside or the thermal camera might have been calibrating through the hardware and found it odd that all it was seeing was blank.

After a while, you will see that the temperatures start to rise at about the 150,000 frame mark. This was because at that time, it was already 6am in the morning and the sun started to heat up the room.

Somewhere near the end of the 8-hour idle test, the temperatures immediately dropped and became noisy. This was actually do to a mistake done during test, however it became a fortunate mistake. The mistake was that the electric fan nearby was turned on. Although the thermal sensor was under a box, the box had two tiny holes on one side. We thought that since the holes were facing away from the electric fan that the readings would not get affected, but it did. From this, it became clear that even the slightest wind

could affect the readings a lot. Another finding was that the wind noise brought the temperature readings down. Looking at how the temperature readings were supposed to go without noise, the noisy readings appear to never go higher than the maximum possible. So this concluded that wind noise is a negative noise and thus, reasoning out why we take the maximum instead of the mean during denoising stage.

Below shows a one hour snippet of the 8-hour idle test taken when the ambient temperature was nearly constant.

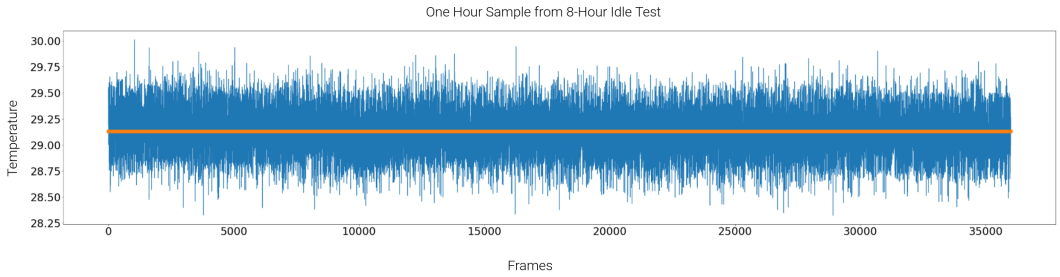


Figure 4.23: One Hour Sample From 8-Hour Idle Test

Comparing this to the tests done in Figure 4.19, There was hardly any wind noise present, making it perfect to measure the isolated sensor noise.

4.2.3.2 Dataset Analysis

The raw dataset gathered, just like the open 1 hour test, had a lot of wind noise, despite making sure to turn off the electric fan while gathering data. Since taking the maximum in 100 frames is insufficient in removing the wind noise, we had to look at all the other tests taken with the same ambient temperature.

Below shows the result of using GMM to level the thermal data.

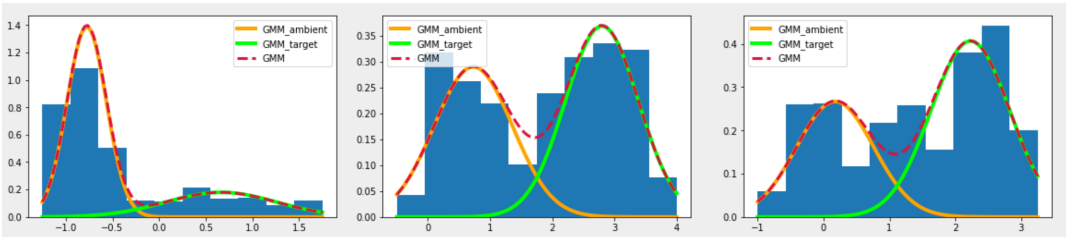


Figure 4.24: Histogram of Difference Between Target and Targetless, Clustered Using GMM

As seen in the figure above, simply getting the largest distribution to get the ambient temperature would most likely fail. This was especially for cases where the target was very close to the camera and the pixels displaying ambient temperature were minimal.

Below shows all the temperature shifts done to the thermal images. The lower the shift, the less noise present.

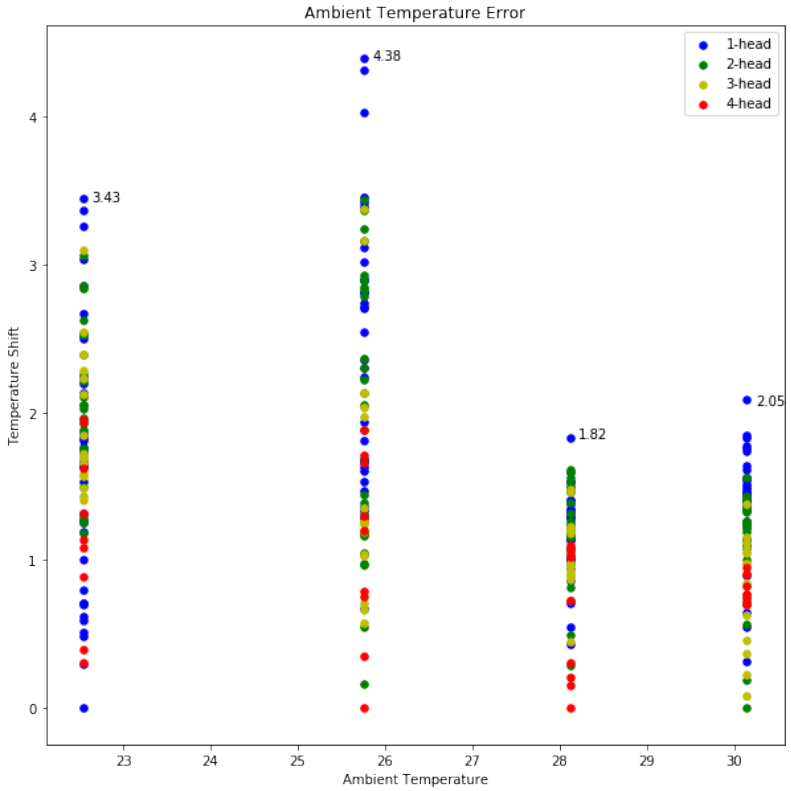


Figure 4.25: Ambient Temperature Error

For more information on the figure above, below are two tables showing variance, mean, and max of different ambient temperatures and target distance measured in head size.

Ambient Temp	Variance	Mean	Max
30.14	0.18	1.14	2.09
28.12	0.12	1.10	1.82
25.75	0.94	1.95	4.39
22.54	0.57	1.76	3.44

Table 4.1: Ambient Error Per Ambient Temperature

Head Size	Variance	Mean	Max
1	0.66	1.64	4.39
2	0.53	1.58	3.44
3	0.44	1.33	3.37
4	0.27	0.92	1.95

Table 4.2: Ambient Error Per Target Distance (Head Size)

Based on the figure and the two tables above, The variance dropped at the head size increased. However, this could have been due to there being less data for increasingly larger head sizes. Recall that head size 1 has 36 samples, 2 has 25, 3 has 16, and 4 has 9.

In terms of ambient temperature, it is evident that colder temperatures have more varied temperature shifts. This could simply be due to it being colder. However, we believe that the cause of the higher variance is due to inconsistent room temperature. These cooler tests were conducted in an air-conditioned room during the summer. The coldest being conducted at night, and the second coldest conducted during the day. The coldest one was less varied than the second coldest most likely because when the room was at the coldest, the temperature could not go any lower and reached ”steady state” or ”terminal temperature”. The aircon used to cool the room does not have any control system to monitor the room temperature but simply blows as much cold air as it can possibly can. During the second coldest, the outside day temperature was most likely affecting the room temperature and caused the temperature to vary greatly. The same could be said with the two hotter temperatures. On the second hottest temperature, the test was taken around 7pm whereas the hottest was taken late in the morning. Temperatures at night were less likely to change than near the peak of noon.

There was another test taken during the afternoon, but the temperatures were so severely hot, the target temperature disappeared among the ambient temperature. The test was aborted and unused due to that reason. From this, we recommend taking temperatures only when the ambient temperature are less than $30°C$.

Below shows the result of GMM levelling for the calibrated dataset.

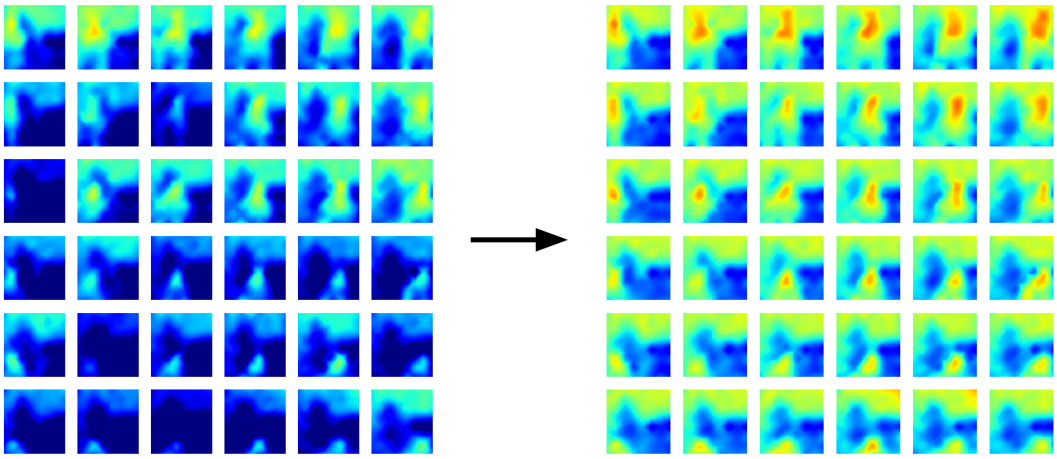


Figure 4.26: Wind Noise Removed using GMM

Finally, below shows the result of a few samples from the final calibrated dataset, ordered with descending ambient temperatures vertically and with descending target distance horizontally.

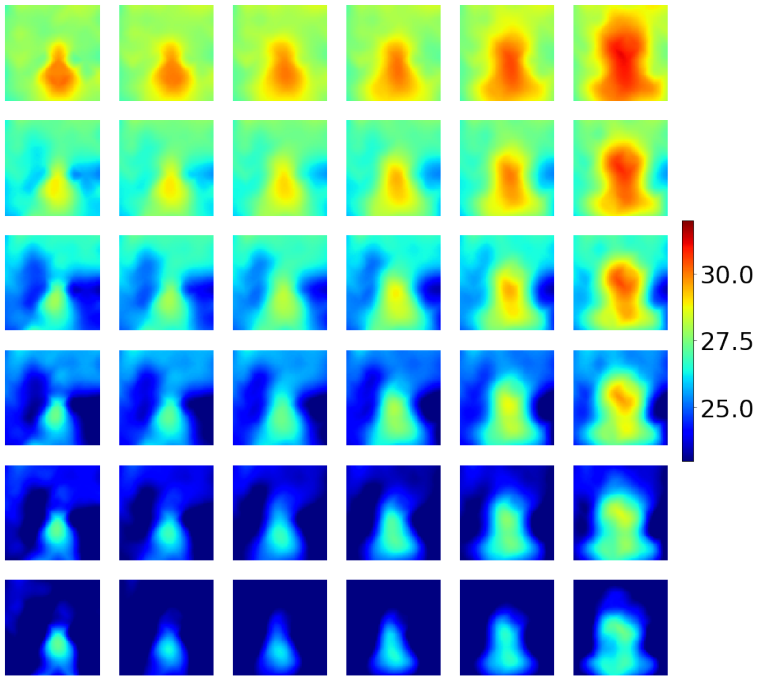


Figure 4.27: Samples From Calibrated Dataset

After making the calibrated dataset, we further analyzed how well it performed in accuracy and in precision. The first figures below shows how the skin temperature varied based on the ambient and target distance and the second shows the maximum offset of skin temperature compared to the average. while the first figure below evaluates accuracy, the second evaluates precision.

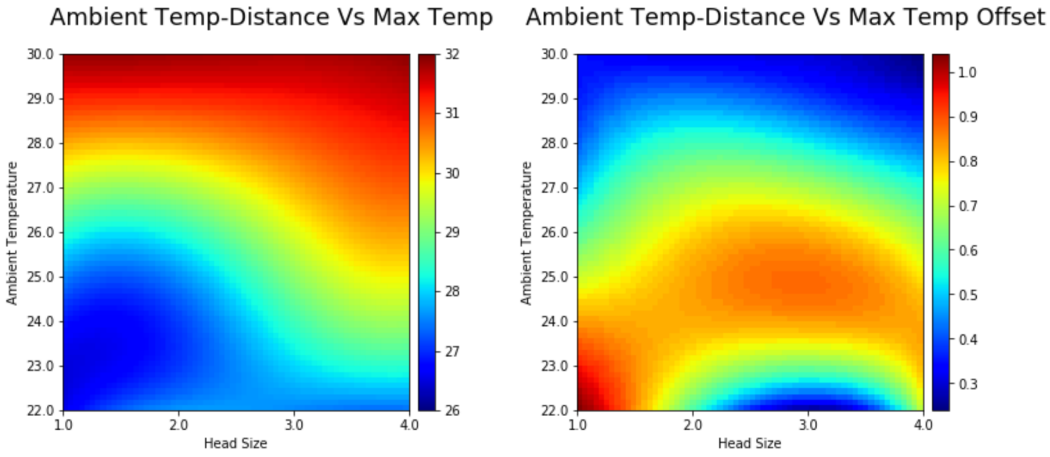


Figure 4.28: Ambient-Distance Vs Skin Temperature and Offset (Max Temp)

As seen in the figures above, hotter ambient temperatures result to hotter skin temperature readings, and more distant targets result to colder skin temperature readings. Note that the actual skin temperature is

not changing but the rather the thermal camera readings are off based on the ambient temperature and target distance.

In terms of precision, the higher the temperature, the more precise the readings are. For som reason, readings taken with target head size of 3 at he coldest ambient temperatures showed a lot of precision. This might have been a fluke though the 2-head and 4-head target distance was also more precise at the coldest temperature than temperatures slightly higher.

Below are two line charts that evaluate skin temperature at different ambient temperatures and target distance. These are similar to the previous figure on the left.

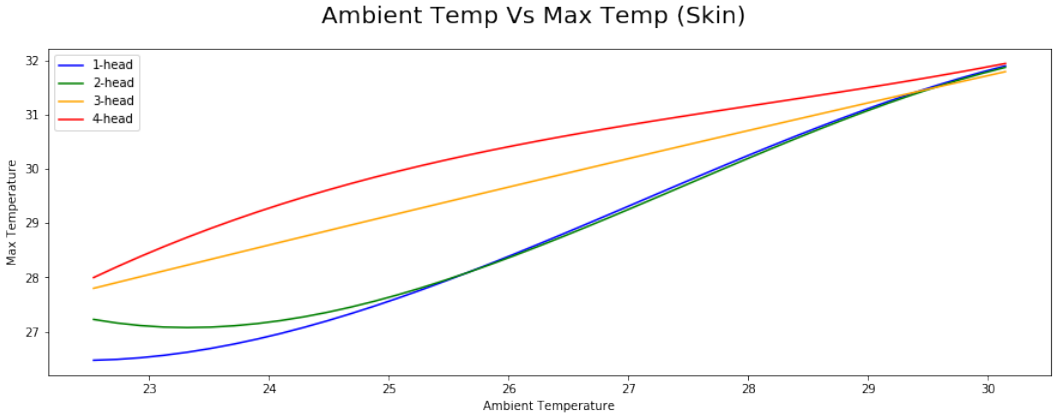


Figure 4.29: Ambient Vs Skin Temperature (Max Temp)

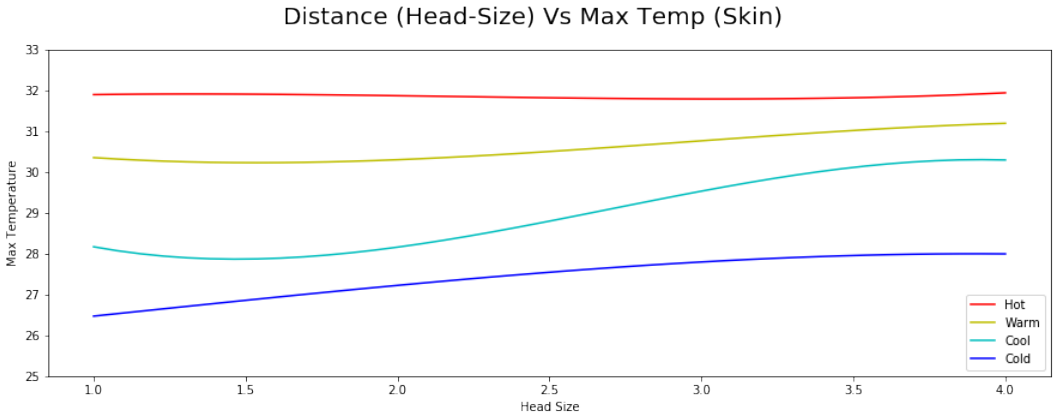


Figure 4.30: Distance Vs Skin Temperature (Max Temp)

These two charts above help demonstrate the bicubic interpolation process done for body temperature prediction. Four polynomials were constructed, similar to that seen in Figure. 4.30. The only difference is while the polynomials in the figure show the max temperature, the body temperature prediction polynomials show the shift in skin temperature to body temperature, which is essentially $36.5°C$ minus the polynomial y-values seen in Figure 4.30. Getting four points from the four polynomials by plugging in the input target distance will produce another polynomial as very similar to those in Figure 4.29

The table below are the actual values used in the bicubic interpolation process to predict body temperature. The output of the bicubic interpolation is added to the skin temperature readings to come up with the estimated body temperature.

Ambient Temperature	1-Head	2-Head	3-Head	4-Head
30.14	4.60	4.63	4.70	4.55
28.12	6.05	6.10	5.64	5.23
25.75	8.33	8.34	6.97	6.20
22.54	10.02	9.26	8.69	8.49

Table 4.3: Skin to Body Temperature Conversion

4.2.4 CNN Evaluation

The figures below show a sample synthesis of skin temperature increase, the output of merging two thermal images together and the 16 distance x 16 ambient temperature interpolated thermal image production.

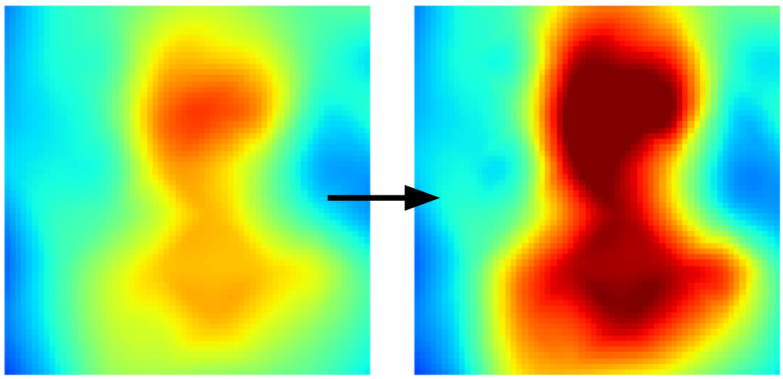


Figure 4.31: Synthesize Fever

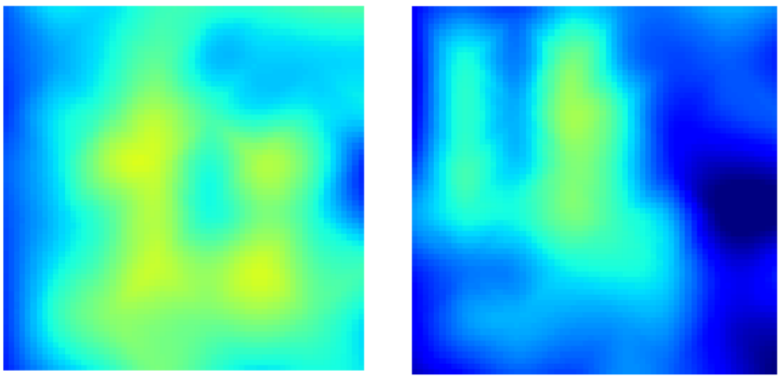


Figure 4.32: Merge Multiple Targets

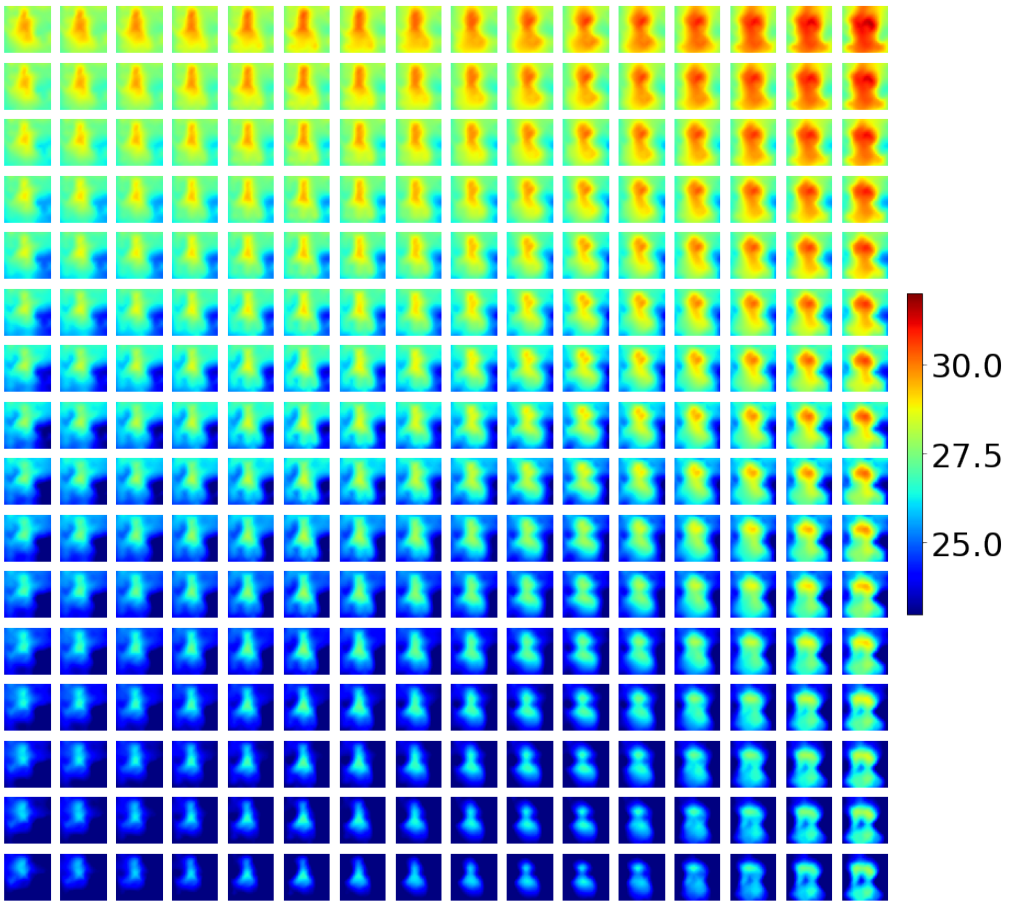


Figure 4.33: Interpolated Thermal Images

After data synthesis, the dataset was fed into the two CNN: unsupervised and supervised. The unsupervised took roughly 4 hours to complete while the supervised took roughly 7 hours to complete. There were several other trainings done but the ones seen below are the latest ones for each. To evaluate the process, the model accuracy and model loss were documented per each epoch.

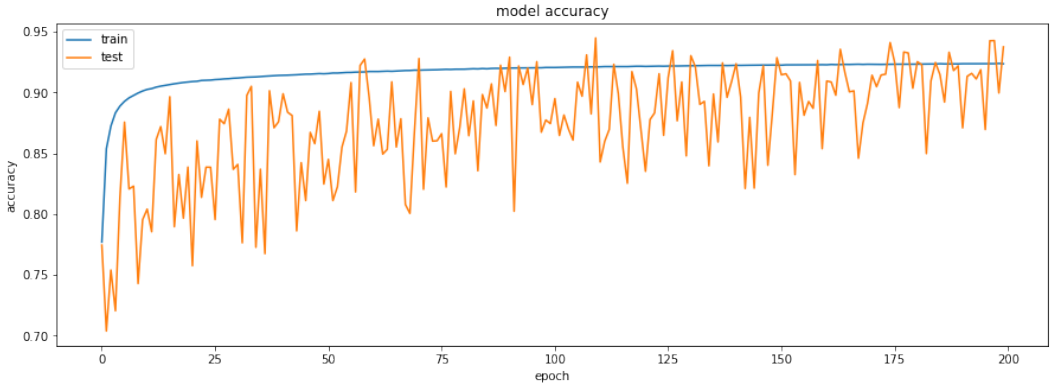


Figure 4.34: Unsupervised (Single Target) Accuracy

Figure 4.35: Unsupervised (Single Target) Loss

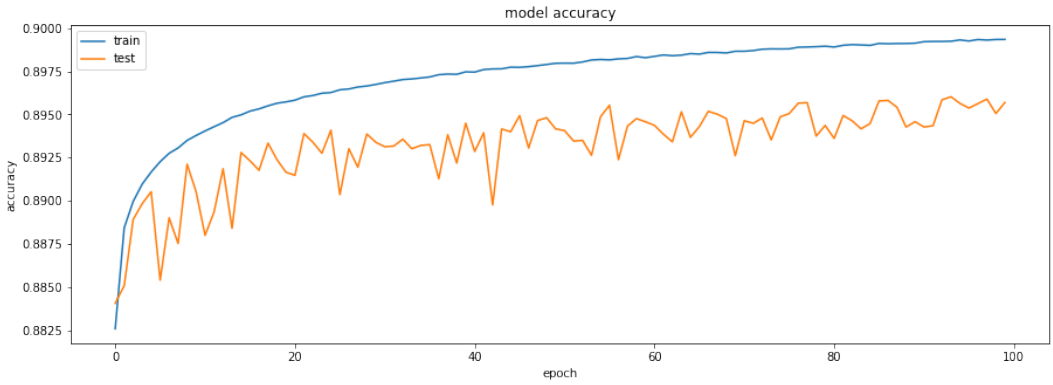


Figure 4.36: Supervised (Multi Target) Accuracy

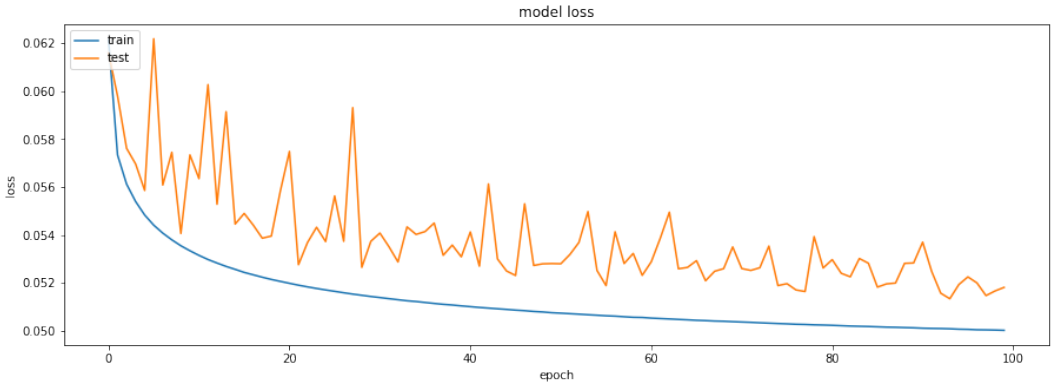


Figure 4.37: Supervised (Multi Target) Loss

Based on the results, The unsupervised model was very noisy in the beginning but gradually became less noisy overtime. The supervised model had a slight overfitting problem but nothing too alarming as it was only off by about 0.5%. Despite the lack of labels, Unsupervised learning came out to be more accurate than the supervised learning. It should be also noted that the unsupervised model outputs 5 parameters while the supervised only outputs 2, which should have made it harder for unsupervised learning to get all outputs correct. However, the unsupervised model also had a huge advantage of being trained for single targets. Thermal images with multiple targets are a lot harder to find patterns and is most likely the reason why the supervised model performed worse.

Before evaluating the CNN models' output error, we made the statistical predictor on finding the skin and ambient temperature used to evaluated the accuracy performance of the CNN models. Below shows the raw scattered data in blue and the fitted line in red. This line was used to convert the minimum and maximum temperatures to Ambient and skin temperatures respectively.

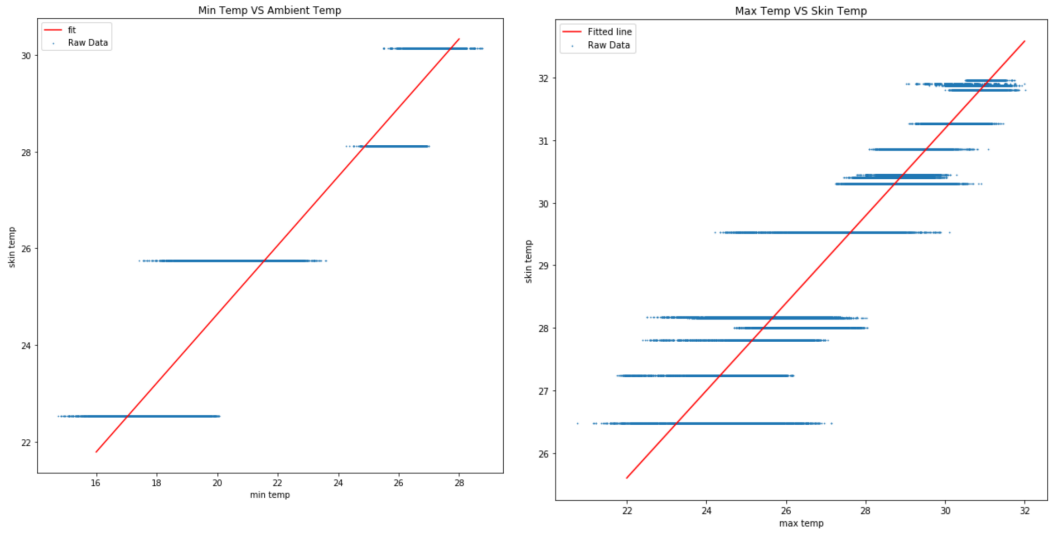


Figure 4.38: Minimum VS Ambient Temperature, Maximum VS Skin Temperature

Both scatter plots have a correlation of about 98%, which makes fitting a straight line justifiable. Just like the unsupervised model, the statistical approach can only be used on single targets.

Below shows a set of 3 distributions for each method. The skin and ambient temperature errors were based on the direct output of each method, while the body temperature error was based on body temperature predictor discussed earlier with the outputs of each method used as the inputs.

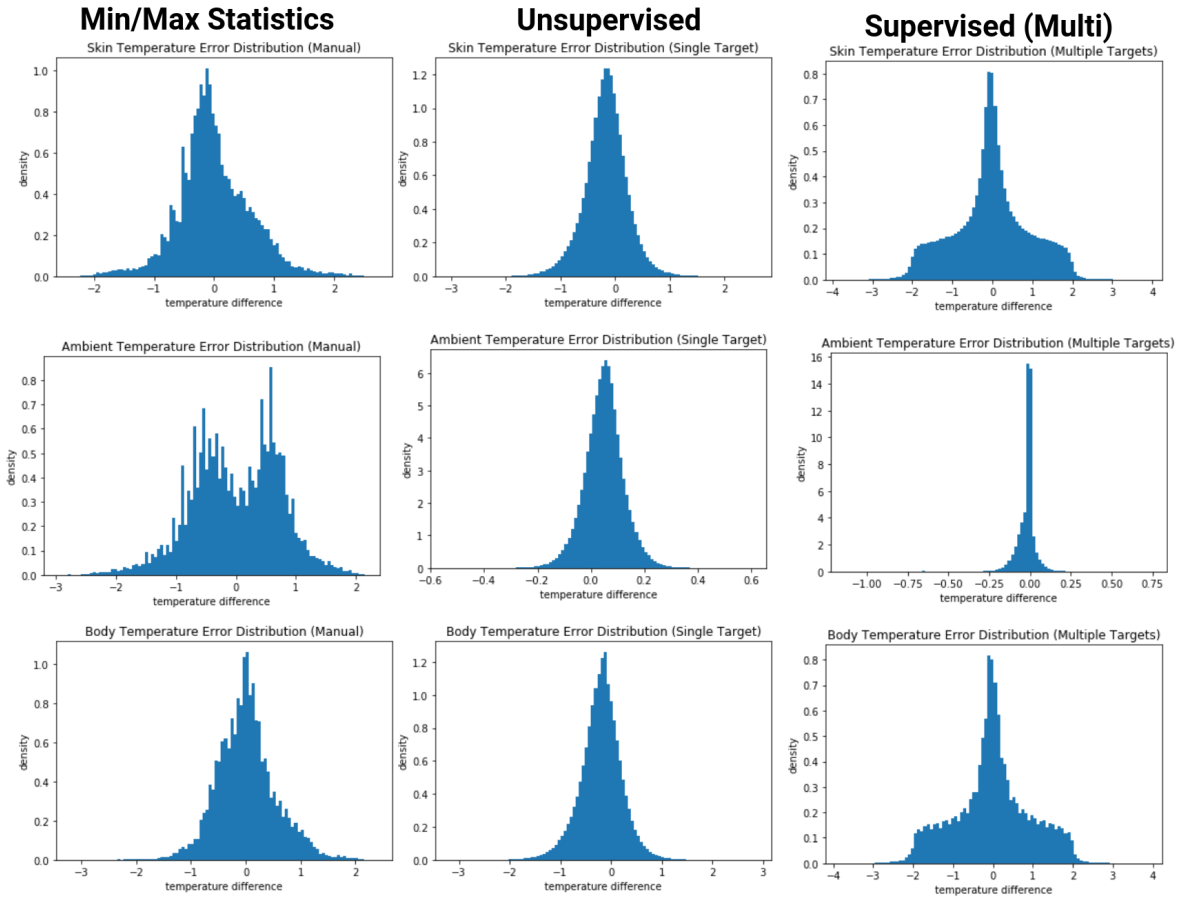


Figure 4.39: Error Distributions on Skin, Ambient, and Body Temperatures of 3 Different Methods

Based on the results, the supervised method performed really well and produced a seemingly perfect Gaussian distribution for all errors. Compared to the statistical approach, unsupervised learning performed more accurately and cleaner with a 0.2 error difference, which justifies the use of CNN over pure statistics. The supervised learning, on the other hand, performed the worse ranging $+/-1.9°C$ 95% of the time. However, this model is the only one that can work with multiple targets. Therefore, if it is necessary to work with multiple targets, the unsupervised CNN approach is the only way. But if accuracy is the priority, the unsupervised CNN approach would be much better.

4.2.5 Final Results

We used the face detection from Adam Geitgey's Face recognition library which works well even on a raspberry pi. This face detection was needed to calculate the targets present, as well as their distance based on the head size and their position. For better user interface, the 8x8 thermal image was upscaled using bicubic interpolation.

The figure below demonstrates how the raw 8x8 thermal readings embedded in green were able to properly convert to body temperature at different ambient temperatures and target distances seen on the left of each image.

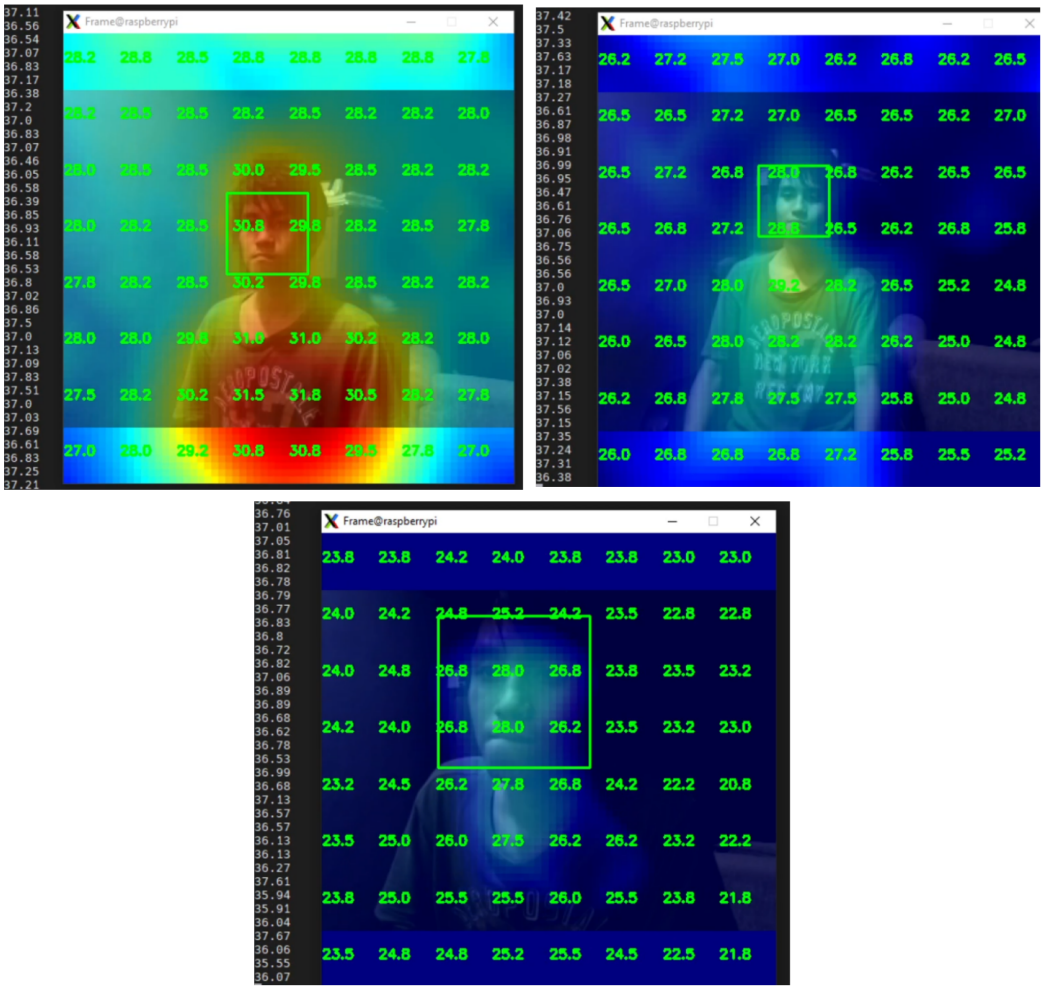


Figure 4.40: Ambient Temperature and Target Distance Calibration Test

the first image above shows the thermal system working on a warm room, the next image shows the thermal system working in a cool room, and the last image shows the thermal system working in a colder room at a closer distance. All were able to predict the temperature to be roughly $36.5 - 27°C$.

The figure below show the results of multiple targets, fever detection and freezing detection work-

ing

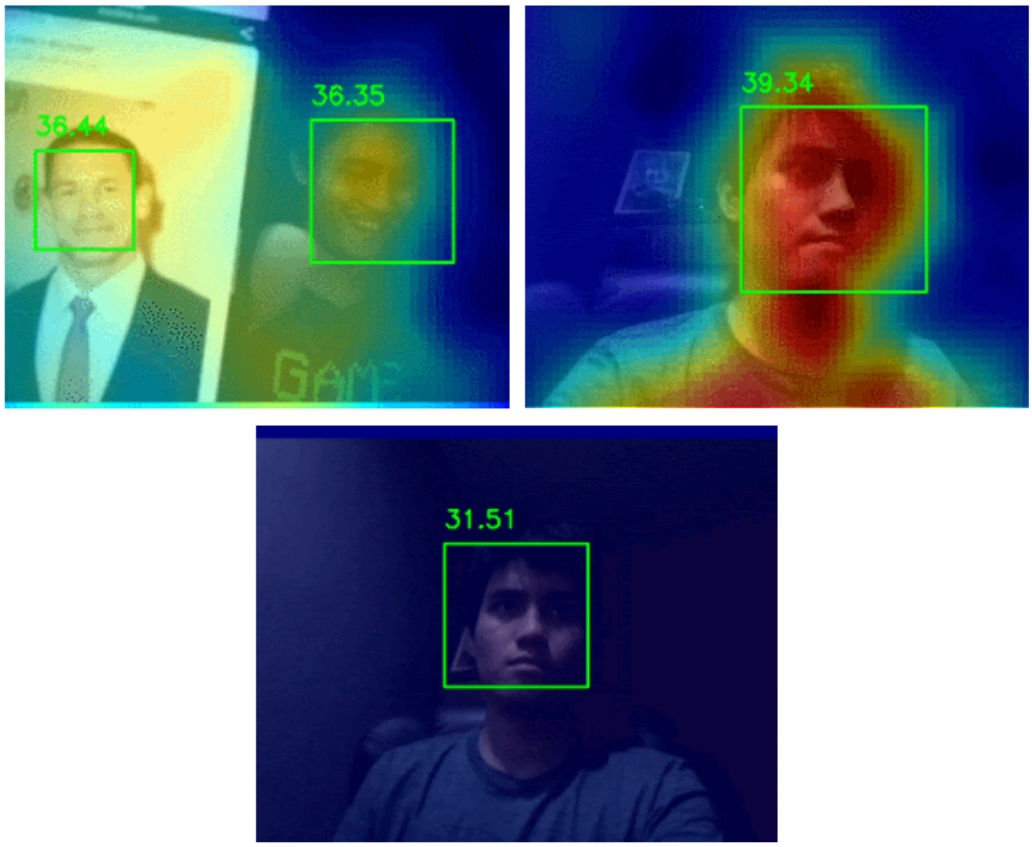


Figure 4.41: Multitarget, Fever, and Freezing Test

The first image above shows the target holding an image of John Cena displayed on a phone. Since the CNN was not trained on phones, it thought John Cena was real and gave him a legitimate thermal reading. To test that the thermal readings was not just spitting out safe temperatures, the ”fever” test was performed. This ”fever” is was synthesized just like the synthesized dataset to make the target appear fake. As seen by the second image above, the thermal imaging system was successful in determining a ”sick” person. Lastly, to test the other end of the spectrum, the thermal image was set to $0°C$ and the this resulted a hypothermia level temperature reading. This demonstrates that the CNN was able to properly asses that cooler faces mean cooler body temperatures and hotter faces mean hotter body temperature, while taking into consideration the ambient temperature and the target distance.

4.2.6 Conclusion

Based on the experiments and tests done, we have concluded that improving the accuracy of cheap thermal cameras, like the AMG8833, is possible and can potentially save a lot of money when compared to using other thermal cameras out there.

One of the things to note is that the current state of the thermal system developed did not reach

the medical standards. The Temperature readings are only confirmed to be correct using the datasets but was not properly evaluated in the actual tests. This is due to lack of more expensive equipment. Ideally, there should be a medical certified thermal camera to compare and contrast with the cheap thermal camera. However, such equipment was not readily available for use. With a medical certified thermal camera with higher resolution, a stronger A.I. could have been trained that can upscale the 8x8 resolution to a much higher resolution, without appearing blurred. Such could be done by either using powerful deep-learning algorithms for image to image generation like Generative Adversarial Networks (GANs) [82] and its variants like Pix2Pix [83] and Cycle-GAN [84]. The stronger thermal camera will be used as the output for training while the weaker thermal camera will be used as the input for training.

Another way this could be improved is through more actual data instead of synthesizing. If the data gathering was not limited and Less included actual raw data with multiple targets and different body temperature, multiple target estimation accuracy would most likely improve.

Given that there are still many ways to improve the thermal system given better equipment and facilities, there is a good possibility that the use of a cheap thermal camera can one day be used as a medically accurate way of detecting fever.

4.3 Accuracy - Calibration through introducing a blackbody reference

One of the ways to achieve better accuracy is through the use of a blackbody. A blackbody is an object that absorbs all incident electromagnetic radiation. It is this property that allows thermal imagers to be calibrated. Ideal characteristics of blackbodies include an emissivity of 1.0, non-reflective surface and adjustable temperature. Emissivity of a blackbody is often attributed to the material it is made of. Attaining an emissivity that is approximately equal to 1.0 is very difficult using common materials. However, the emissivity of an object can be improved by constructing the material into a cone-shape.[85]

Despite existing solutions allowing only manufacturers to perform hardware calibration on their sensors, introducing a blackbody within the field of view of the thermal imager will allow us to calibrate our thermal readings. This would be possible by doing the following:

- Make a blackbody and implement a PID algorithm using RPi and cheap hardware components
- Compare measurements of the baseline sensor of the blackbody and the thermal camera and calibrate the temperature reading based on measurements

4.3.1 Construction of Blackbody

For the blackbody, construction of 2 units is required. Namely the heater unit and the temperature control unit. Below shows the components of each unit.

- Heater Unit
  - Radiator (Cone)
    - 125mm x 30mm (height x radius of cone)

    - Paint black
  - Electric Heater
    - 0.1m Nicrhome Wire (0.17mm diameter)
    - 0.04m Fiberglass Wire Sleeve
  - Power Source
    - 12V 2A DC Source
    - DC Connector
- Temperature Control Unit
  - Raspberry Pi 3
  - DS18B20 thermometer
  - MOSFET: IRFZ44N

For the heater unit, the cone is constructed out of a metal sheet with the dimensions of the cone being 125mm x 30mm (height x radius). Based on the equations shown below[85] these dimension theoretically improve the rated emissivity of the material (ξ=0.9) to approximately ξ=0.9747 however, the proponents do not have the proper equipment to verify this. The cone is closed on all sides and is hollow inside. More details on the dimension are shown in Figure 4.42.

$$\epsilon_{app} = \frac{\epsilon}{1 - (1 - \epsilon)F_{2,2}}$$

where:

$$F_{2,2} = 1 - \frac{R}{\sqrt{H^2 - R^2}}$$

$R -$ radius of the cone bottom

$H -$ height of the cone

$\epsilon -$ emissivity of the inner surface of the cone

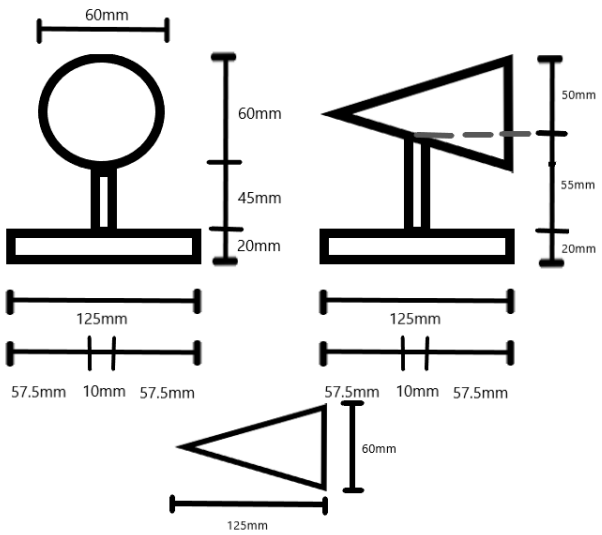


Figure 4.42: Dimensions of Radiator

The fiberglass wire sleeve will be cut open allowing it to partially wrap around the blackbody which will then be wrapped around by the nichrome wire. We then attach the DS18B20 on top of the blackbody. Figure 4.43 shows this setup.

Figure 4.43: Blackbody, fiberglass wire sleeve, nichrome wire, DS18B20

For the temperature control unit, the DS18B20 is connected to the Raspberry Pi . It should also be noted that a 4.7kΩ resistor is connected between the Vcc pin and the Data pin of the DS18B20. The Gate of the IRFZ44N is connected to the is connected to physical pin 12 (GPIO18) of the raspberry pi. Physical pin 12 is the pin that we will use to send our PWM signal to the rest of the circuit. A more detailed schematic can be seen in Figure 4.44.

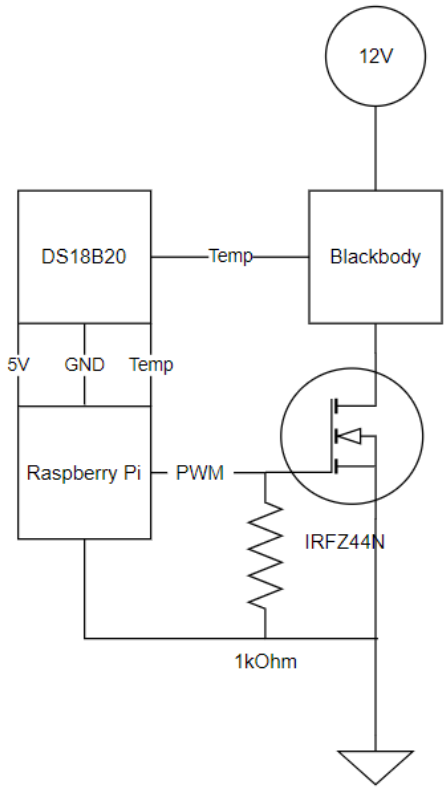


Figure 4.44: Blackbody Schematic

4.3.2 PID Tuning Process

For the PID tuning process a method called Empirical Gain Tuning was used. First, we would introduce a P constant. We would then gather data, increment, and gather data until no desirable change happen. We then introduce an I constant then gather data, increment, then gather data until no desirable change happens. We then try to verify if our results are the best so far by changing only one constant at a time for data gathering. We then repeat the process by introducing a D control and also making only one change at a time to verify our results. It should also be noted that one data gathering last for about 50 minutes meaning we let the PID algorithm run 50 minutes after which we plot the measured temperature. The sampling rate is approximately 1 second. Itemized steps can be seen below and a flowchart as well in Figure 4.45

1. Set a value to the P constant and 0 to the I and D constant and begin data gathering.
2. Increase P by a factor of 10 and gather data. Continue to increase by a factor of 10 until no desirable changes happen between current test and previous iteration. (ex. previous - 1000, current - 10000)
3. Choose the constant of the previous iteration and play around from there. (ex. 1000)
4. Only change one constant at a time.
5. Introduce I constant and repeat step 2, 3, and 4 for I constant.

6. Introduce D constant and repeat step 2, 3, and 4 for D constant.
7. Repeat PID constants to verify results.

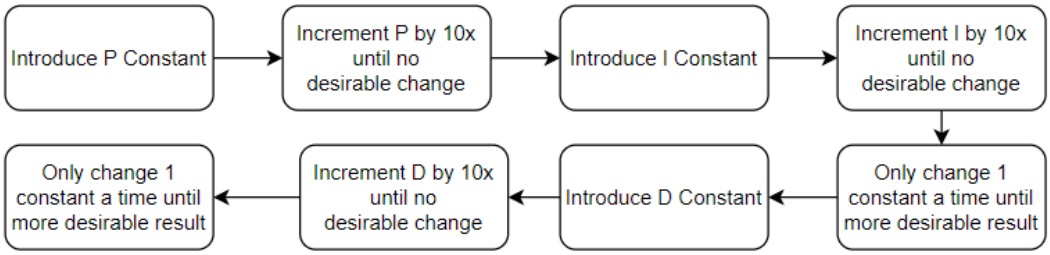


Figure 4.45: Empirical Gain Tuning Flowchart

Furthermore a snippet of the of the python code where the constants are changed can be seen below in Figure 4.46

```
while True:
    temp = sensor.get_temperature()
    stop = time.time()
    tup = (temp, duty_cycle, stop-start)
    print(temp, duty_cycle, stop-start)
    writer = csv.writer(f)
    writer.writerow(tup)

    diff = set_temp - temp
    sum_diff += diff
    P_control = P*diff
    I_control = I*(sum_diff)
    D_control = D*(diff-diff1)
    diff1 = diff;
    duty_cycle = P_control + I_control + D_control
    if duty_cycle > 100:
        duty_cycle = 100
    elif duty_cycle < 0:
        duty_cycle = 0;
    else :
        continue
```

Figure 4.46: Snippet of Python Code Used

4.3.3 Characterising the Blackbody

We will be basing our temperature reading from the DS18B20 attached to the radiator. We will find the following characteristics:

- Rise Time
  - Time required for the response to rise from 5% to 95% of its final value. This will tell us how long it takes the blackbody to reach a set temperature.
- Settling Time

  - Time required for the temperature of the blackbody to become steady at a set point. It is defined as the time required by the response to reach and steady within 1% of its final value.

- Overshoot
  - Difference between the magnitude of the highest peak of time response and magnitude of the temperature steady state. Maximum overshoot is expressed in term of percentage of steady-state value of the response. As the first peak of response is normally maximum in magnitude, maximum overshoot is simply normalized difference between first peak and steady-state value of a response.

- Steady State Error
  - Defined as the difference between the desired value and the actual value of a system output as time goes to infinity.

4.3.4 Results of Characterising

For the results of the characterizing of the prototype, the best results for a set temp of 35°C can be achieved by using the PID constant of 500, 0, and 0 respectively. Meanwhile, for a set temp of 40°C with PID constants 1000, 5, and 0 respectively yields the best results.

For all the data gathered, the PID constants which showed the best results, for a set temp of 35C the average maximum overshoot is 36.0625°C. The average rise time and settling time is 100 seconds and 1,780 seconds respectively. Now the approximate final value of a set temperature of 35C°C is 34.875°C. sometimes it would fluctuate to 35°C but then it would settle again at 34.875°C. However, for the set temperature of 40C the temperature was not able to settle as seen here. The temperature tended to fluctuate between 39.75°C and 40.25°C. The average maximum overshoot and rise time are 41.75°C and 373 seconds respectively. Shown in Figure 4.47 and Figure 4.48 are summaries of the data gatherings conducted for 35°C and 40°C respectively. Highlighted in yellow are the best resulted described earlier. It should be noted that data shown for the best results for 35°C is a summary of 3 separate data gatherings which showed and confirmed that the results were consistent and repeatable. Meanwhile for the 40°C, data shown below were from 4 separate data gatherings.

35C								
P	I	D	Max Overshoot(C)	Rise Time(s)	Settling Time(s)	Approx Final Value(C)	Fluctuation	Stready State Error
100	0	0	N/A	N/A	135	34.4375	[34.375, 34.5]	[-0.625,-0.5]
1000	0	0	36.5	64	N/A	N/A	[34.8125, 35.5]	N/A
10000	0	0	36.5	64	N/A	N/A	[34.8125, 35.5]	N/A
500	0	0	36.0625	100	1780	34.875	[34.875, 35]	[-0.125,0]
500	1	0	36.3475	66	N/A	N/A	[34.8125, 35.9375]	N/A
500	10	0	37.0625	55	N/A	N/A	[33.625, 37]	N/A

Figure 4.47: Results for set temperature 35°C

40C								
P	I	D	Max Overshoot	Rise Time	Settling Time	Approx Final Value	Fluctuation	Stready State Error
100	0	0	40.0625	581	N/A	N/A	[38.875, 39.25]	N/A
1000	0	0	40.3125	657	N/A	N/A	[39.375, 40.3125]	N/A
10000	0	0	40.625	370	N/A	N/A	[39.6875, 40.75]	N/A
1000	1	0	41.75	375	N/A	N/A	[39.75, 40.5]	N/A
1000	10	0	49.75	372	N/A	N/A	[48.50.,49.75]	N/A
1000	5	0	41.75	373	N/A	N/A	[39.75, 40.25]	N/A

Figure 4.48: Results for set temperature 40°C

4.3.5 Demonstration of Integrated Setup

An integrated setup where the AMG8833 sensor is used while having the blackbody and people passing by within the field of view of the sensor was not possible due to personal circumstances of the proponents and restrictions during the pandemic. However, a sample of what the setup would have looked like was provided using a Keysight U5855A thermal imager. A sample image shown in Figure 4.49 was taken using the Keysight U5855A thermal imager while having the blackbody and a person passing by within the field of view. It should be noted hear that the blackbody is not connected to the rest of the circuit as there were restrictions and guidelines that the proponent had to follow in the hospital that lent the proponent the thermal imager.

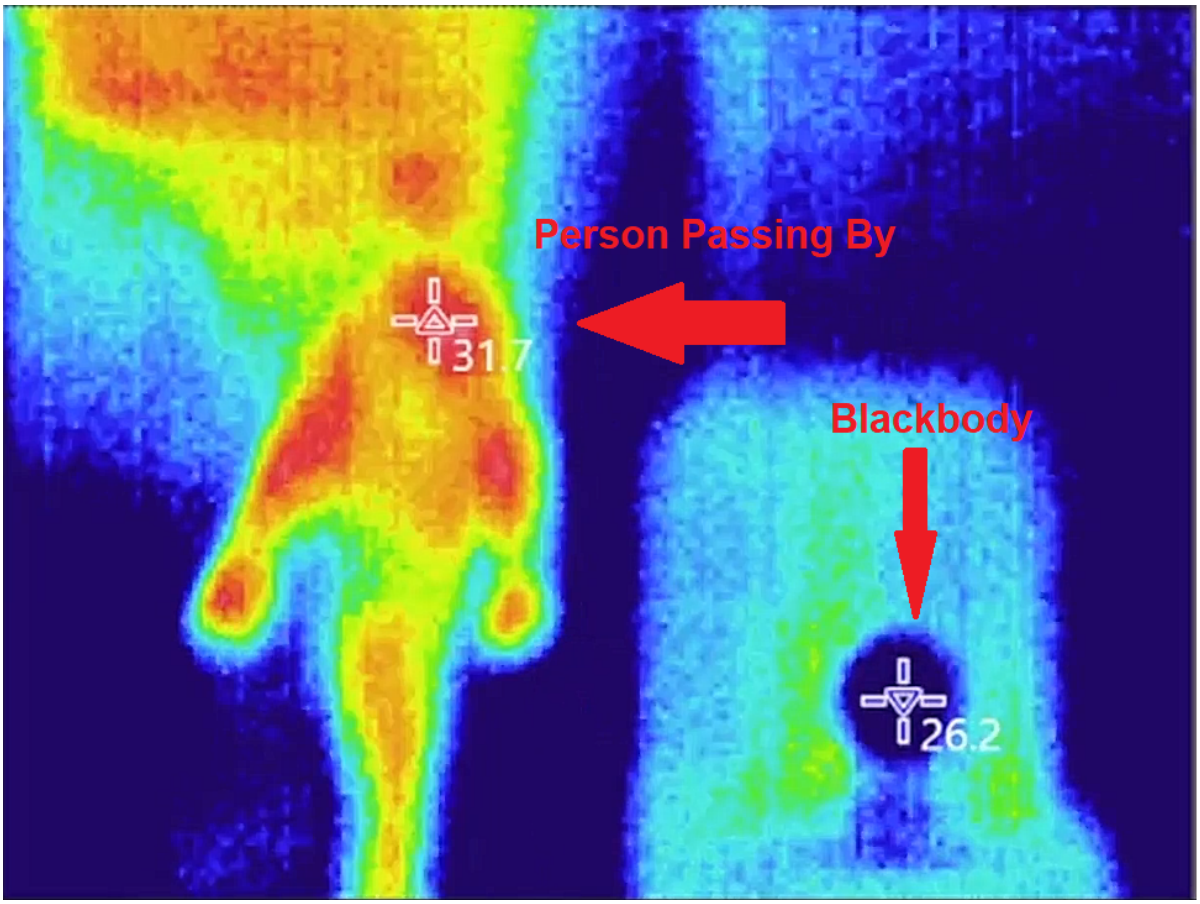


Figure 4.49: Integrated setup with Keysight U5855A, blackbody, and person passing by. °C

4.3.6 Conclusion and Recommendation

This section demonstrates that it is possible to integrate a blackbody into our thermal imaging system proving that this system is hardware expandable. Since the temperature can be set it can be used for calibrating a thermal imager for better accuracy. A point to improve is to integrate the setup with a thermal sensor (e.g. AMG8833) that can be connected to the Raspberry Pi. Also, the paper titled "An inexpensive blackbody model" by M. Fidali, M. Mikulski [85], which the design of our blackbody was based on, presented a blackbody model which is capable of being set to even more various temperatures. The design of their blackbody is however much more difficult to construct given that part of the blackbody is submerged in water.

4.4 Feature Set

In this section we will implement iris recognition as an additional feature that is useful for contact tracing . This will be done by integrating iris recognition to our thermal screening system and will serve to identify the subjects that will be screened by our thermal screening system. The set-up ran on a Raspberry Pi 4 Model B with an ARM CPU running at a maximum clock speed of 1.5 GHz and a Raspberry Pi Camera

module V2. The iris recognition was also tested on a server with an intel CPU running at a maximum clock of 3.4 GHz and an NVIDIA GTX 1080 GPU. Python is used as the programming language which included libraries for PyTorch, OpenCV, NumPy, SciPy, Matplotlib, scikit-image, and PyYAML. The dataset used to test the iris recognition system is the first 250 identities from the CASIA-Iris-V4-Lamp collected by the Chinese Academy of Sciences, Institute of Automation.[86]. The dataset contains images collected using OKI's IRISPASS-h, a handheld iris sensor. A lamp was turned on and off close to the subject to introduce variation due to pupil expansion and contraction under different illumination conditions.

4.4.1 Iris Segmentation

Input images from the dataset are resized and converted into a single channel image or a greyscale image in preparation for the iris segmentation network. The iris segmentation network used is CC-Net architecture[87]. CC-Net is a lightweight biomedical image segementation model with a U-Net structure[88]. In the original paper, CC-Net is shown to be able to retain 95% accuracy while using only 0.1% of the trainable parameters on image segmentation tasks compared to full sized CNN models. The lightweight size of the CC-Net model makes it ideal for running on an embedded device with limited memory or computing resources. The CC-Net network compression scheme was applied to the domain of iris images by a different paper titled, "Open Source Iris Recognition Hardware and Software with Presentation Attack Detection"[89]. The source code used for this project for the implementation of the iris recognition system was also uploaded by one of the authors.[90] The output of the segmentation network was resized to the input size and then converted to boolean format. Figure 4.50 shows a sample image from the CASIA dataset and the output of the iris segmentation network.

Figure 4.50: A sample image from the CASIA-Iris-Lamp dataset (Left) with the corresponding output of the iris segmentation network(Right)

4.4.2 Normalization

Hough Circle transform is applied to obtain circles which corresponds to the iris and the pupil. The output circles is defined by the 3 values. The x and y coordinates of the center of the circle and the radius. Figure 4.51 shows the two circles generated by the Hough circle transform overlayed with the output from the image segmentation network.

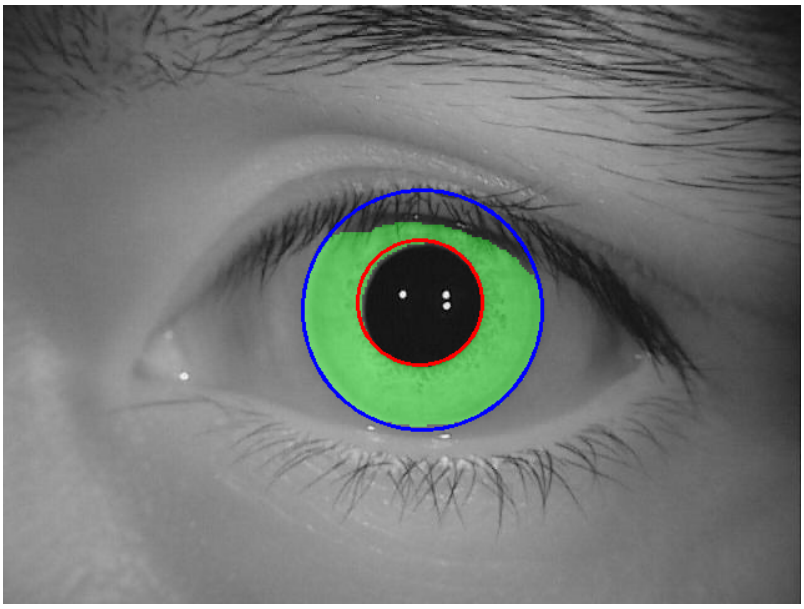


Figure 4.51: The two output circles from the Hough Circle Transform operation overlayed on top of the original input image

After performing circular approximation, Daugman's rubbersheet model is used to transform the area between the iris and the pupil and normalize it into a definite rectangular template for iris patterns[91]. It has an effect of unrolling the circular area covered into rectangular template with a size of 64 x 512 pixels. The rubbersheet model is also applied to the iris segmentation output to create a mask which will be used later in the matching stage to remove the non-iris area covered by the Hough transform output such as eyelids and eyelashes.

4.4.3 Feature Extraction

In order to extract features from the normalized input, the domain-specific BSIF-based iris recognition algorithm is applied[92]. BSIF filters were previously developed using a small set of natural images which serves as general feature extractors for any kind of application. In this algorithm, the general BSIF filters was adapted into the domain of iris images. This is done by using patches from iris images to develop and train the BSIF filters in order to be used for iris recognition. The patches or regions of interest from the iris images used to develop the BSIF filters was chosen using the opinion and behaviour of human volunteers. The computation of filters is done with the scikit-learn Fast ICA implementation. For our project, the filters are stored as a mat file which is loaded up during the start of the program. Seven filters were used with a size of 15x15. Each filter is convolved over the normalized input independently. After feature extraction, the output is a boolean array with a shape of 64 x 512 x 7.

4.4.4 Matching

Given two iris codes which are outputs from the feature extraction stage, they are compared by calculating the fractional Hamming distance of the non-occluded iris codes. The hamming distance is

computed by taking the output of an exclusive OR operation on two iris codes and then taking the sum of the elements. The output mask from the normalization stage is then used to remove the areas of the iris codes which corresponds to non-iris regions such as eyelids or eyelashes. Then it is divided by the intersection of the mask of the two iris codes in order to normalize the Hamming distance to values between 0 and 1. First, this process is repeated multiple times with each time shifting the second iris code by one. This is done from the second iris code shifted 16 times to the left up to a shift of 16 times to the right. This is done in order to account for possible head tilt from the input. The lowest Hamming distance calculated is chosen as the score between two iris codes.

Using images from the CASIA dataset, 88,672 pairs of images that belongs to the same person were used to compute the scores generated for matches. One image from each identity were used to choose 27,029 pairs of iris images that do not belong to the same person to compute the scores generated for nonmatches. Figure 4.52 shows the histogram or the score distribution for the matches and non matches. Figure 4.53 plots the two histograms in one plot to show the intersection between the score distributions.

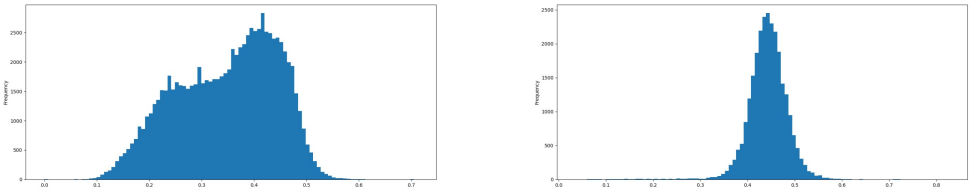


Figure 4.52: The score distribution for matches (Left) and the score distribution for nonmatches (Right)

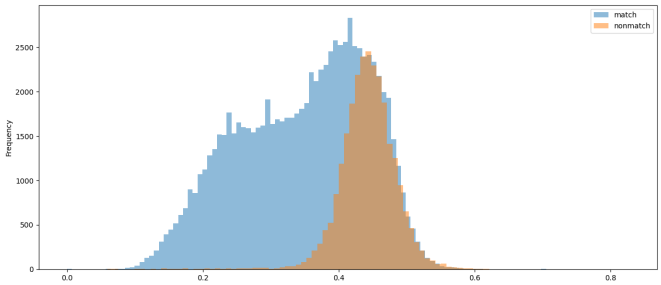


Figure 4.53: The score distribution for both matches and nonmatches

The lower the score the more likely the pair of iris images belong to the same person. The average score for matches is 0.348 with a standard deviation of 0.095. The average score for nonmatches is 0.442 with a standard deviation of 0.042. If a threshold will be set at 0.35, this will minimize the false match rate to 1.58% but it will correspond to a false non-match rate of 52.9%. Based on the calculated standard deviation for non-matches, the threshold can be increased to 0.4. This will increase the false match rate to 11.14% but it will decrease the false non-match rate to 34.9%.

Images taken from the Raspberry Pi camera module were also tested. The images were taken as close as possible to the camera. While the iris recognition system can accept and processes the input image

without errors, the scores generated during the matching stage cannot distinguish iris images from different people and will always correspond to a score less than 0.35 or a score which correspond to a match. Figure 4.54 shows the sample output of the iris recognition system before the matching process.

Figure 4.54: A sample image taken using the Rpi Camera module (Left) and the output of the segmentation and normalization stages of the iris recognition system(Right)

4.4.5 Performance

The iris recognition system is executed and benchmarked on the Raspberry Pi to determine the time it takes for each stage to finish. The process is repeated by executing the iris recognition system on a GPU server. For the segmentation,normalization, and feature extraction stage, the time presented corresponds to the time it takes to process one iris image. For the matching stage, the time presented is the time it takes to process a pair of iris codes. Table 4.4 show the runtime for the Raspberry Pi and the GPU server as well as the speedup between the two devices.

Iris Recognition Benchmark			
	Raspberry Pi	GPU Server	Speedup
Segmentation	0.771 s	0.031 s	24.87
Normalization	0.970 s	0.275 s	3.52
Feature Extraction	0.596 s	0.099 s	6.02
Matching	0.27 s	0.05 s	5.4

Table 4.4: Runtime for each stage of the iris recognition system in seconds and the speedup between the two devices

As seen in Table 4.4, there is an increase in performance in the iris recognition system if the process will be transferred to a more powerful device. While the speedup for the Normalization, Feature Extraction, and Matching stage corresponds to the increased CPU clock speed of the server to the Raspberry Pi, we can observe a massive speedup for the iris segmentation stage because the iris segmentation stage uses PyTorch which can utilize the GPU resources of a device.

4.4.6 Conclusion and Recommendations

In this section, we demonstrated the ability of our project execute a function not limited to thermal screening but can be used for recording purposes which is useful for contact tracing. Our project is capable of taking images and processing them for iris recognition. It is also possible to limit the function of the embedded device to taking input images and to transfer the process of iris recognition to a GPU-powered device if necessary.

An improvement that can be applied to the project is the addition of a dedicated iris scanner or a near-infrared camera capable of taking higher quality iris images. It is highly likely that the images taken using the Raspberry Pi camera module is not detailed enough for the feature extraction stage to properly represent the input iris image. Another recommendation would be to develop a system which can store and trace back the output of the feature extraction stage for offline storage for images stored in the device such that the first three stages of iris recognition would not be repeated each time a new input image is introduced.

4.5 Expandability

In this section, we exhibited the expandability of our system by demonstrating the ability of our system to deploy virtualized functions and chaining these said functions. The Set-up ran on a Raspberry Pi 4 Model B with 8GB of RAM. We used 4 virtualization platforms: Docker, LXC (Linux Containers), KVM (kernel-based virtual machine), and VMWare Esxi. The virtual functions demonstrated were video streaming and compression. The application that was used was VLC because of its ability to perform these two functions. The containers and VMs were ran on an Ubuntu enviornment.

The first implementation containerized VLC in Docker. There were two containers, container-1 playbacked a 20 minute video file which was sent to container-2 through HTTP with an H.264 video codec and an MP3 audio codec, container-2 then compressed the played video to a preset vlc profile Video for iPhone. These containers will connect and communicate using a bridge network. This demonstration was also recreated with LXC, KVM, and VMWare Esxi. In the case of KVM and VMWare, VLC ran on Virtual Machines rather than containers. These platforms provide tools that allowed the tracking and recording of the CPU and memory usage.

4.5.1 Docker Implementation

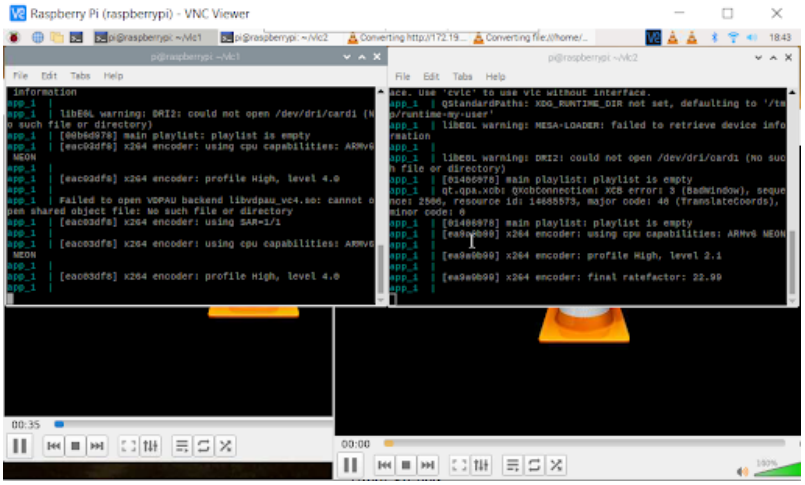


Figure 4.55: Docker Setup

Figure 4.55 shows the working setup of the containerized VLCs with the use of Docker. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings [93]. Docker images of popular x86 applications are usually available in docker-hub however the Raspberry Pi 4 is an ARM device, with the difference of computer architecture we had to create our own Docker image. The docker container was built by creating an Ubuntu environment then installing vlc in that environment. A directory was mounted to the container for specific file access. The container was also granted X11 access so that it can display a GUI (Graphical User Interface). A non-root user was also applied as VLC cannot be run as root.A bridge network with Docker was then created; the containers were able to communicate with each other. With the robust capabilities of VLC it has the ability to stream the file through HTTP to another container that will compress it.

4.5.2 LXC Implementation

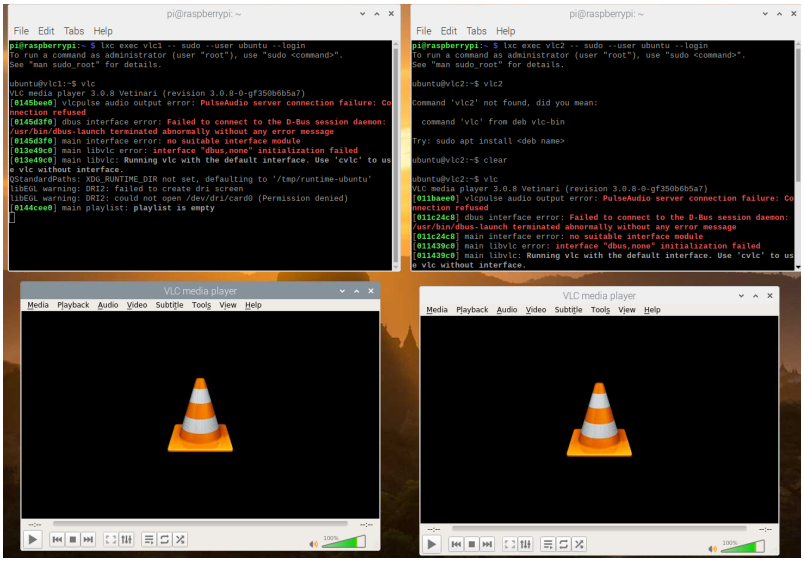


Figure 4.56: LXC Setup

Figure 4.56 shows the working setup of the containerized VLCs with the use of LXC. LXC is container management interface built for linux users. Creating the VLC linux container only needed the setup of the environment, the application being installed, X11 access, and the mounting of necessary directories. The setting up of the containers automatically connects the containers through a bridge network. As with Docker, VLC performs the necessary functions for demonstration.

4.5.3 KVM Implementation

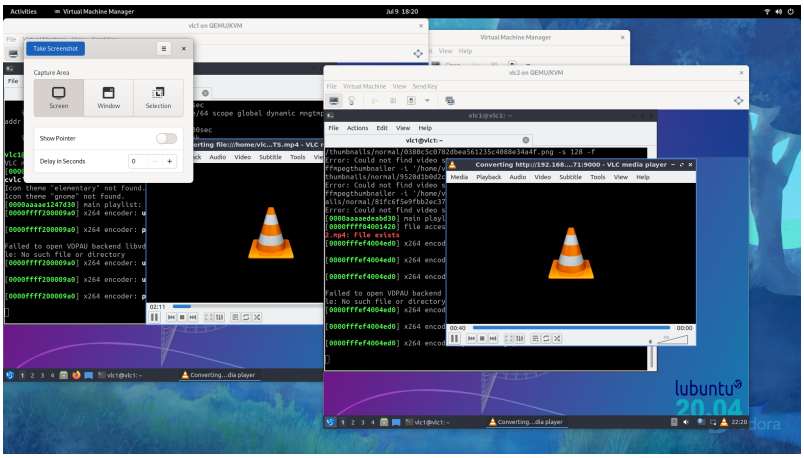


Figure 4.57: KVM Setup

Figure 4.57 shows the working setup of an Ubuntu Virtual Machine running VLC through KVM. The Raspberry Pi 4 does not support hardware virtualization as /proc/cpuinfo doesn't display any flags indicating it does. This does not mean that the Raspberry Pi cannot run KVM but it will not run optimally. Trying to run KVM in Raspbian OS also leads to the VMs created experiencing kernel panic. However by running the Raspberry Pi with Fedora Workstation as the Operating System it was able to run KVM. For this implementation, we still had to create a bridge network interfaced by an ethernet adapter to allow the KVM VMs communication within the local access network.

4.5.4 VMWare Implementation

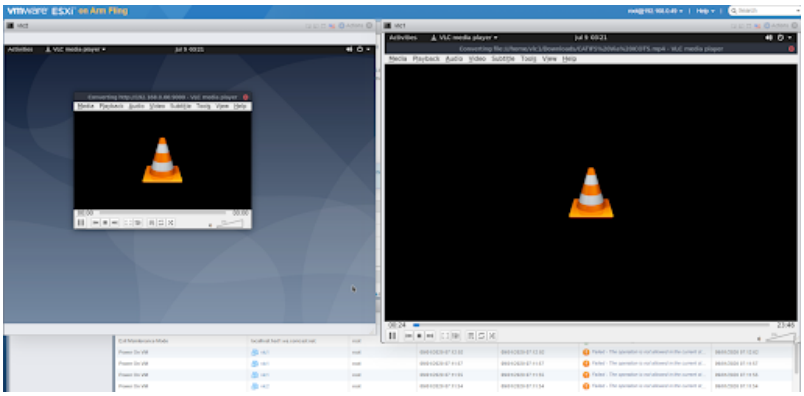


Figure 4.58: VMWare Setup

Figure 4.57 shows the working setup of an Ubuntu Virtual Machine running VLC through VMWare. VMWare is a bare-metal server that is commonly used as an enterprise solution. This implementation dedicates the device to be run as a server hosting virtual machines. VMWare also allows ready communication

between the VMs created within the server. It was also easy to setup the VM with VMWare as one only needs to traverse the menu to create a VM.

4.5.5 Summary

Average Resource Consumption				
Virtualization Platform	CPU		Memory	
	vlc1	vlc2	vlc1	vlc2
Docker	350%	175%	600 MB	151.6MB
LXC	78.7%	67.3%	633.3 MB	195.9 MB
KVM	268%	104%	1816MB	1388 MB
VMWare	198.52%	132%	2.28 GB	1.99GB

Table 4.5: Average Resource Consumption

As shown in Table 4.5 the LXC implementation displayed the least usage of CPU while the Docker implementation used the most. With regards to memory usage, Docker consumed the most but lxc does not have that much difference. The VMs however used much more because they were running full operating systems. In regards to using the least amount of resources, LXC is the best option for virtualizing functions.

4.5.6 Conclusion and Recommendations

This section provided the ability of an interconnected, open, programmable, and extensible thermal imaging platform with regards to expandability. By virtualizing the functions, we were able to demonstrate that our programmable and interconnected approach to thermal imaging yields greater flexibility through by enabling the dynamic deployment of new and distributed functionality.

An improvement that could be applied to the project would be to perform the tasks fully on the command line without a GUI to lessen resource usage which could be used performing other functions. Another recommendation would be automating the process through scripts to decrease time consumed navigating through menus and options.

4.6 Performance and Scalability

In this section, we will be demonstrating possible modifications on thermal imagers ran on Jetson Nano and their effects on the system's performance and scalablity. We will be using three different approaches in modifying the thermal imager. For programmable, we will be demonstrating the speedup of the thermal imager by using builtin compute resources. For interconnectivity, we will be demonstrating the speedup of the thermal imager by offloading tasks to server. For hardware expandable, we will be demonstrating the ability of the thermal imager to process multiple subjects.

4.6.1 Design Overview

In order to demonstrate the ability of a thermal imager to process multiple subjects, we will be creating a thermal imager on Jetson Nano which can detect multiple faces in real time which will then be

processed in order to be cross referenced with the thermal image and to read the temperature of the subjects given the localized areas.

	Control Group	Experimental Group 1		Experimental Group 2	
Architecture	CPU (Jetson Nano)	CPU (Jetson Nano)	GPU (Jetson Nano)	CPU (Jetson Nano)	GPU (Server)
Functions	- Video encoding and decoding - Face detection - Thermal image quality improvement - Localizing temperature reading sites	- Video encoding and decoding - Localizing temperature reading sites	- Face detection - Thermal image quality improvement	- Video encoding and decoding - Localizing temperature reading sites	- Face detection - Thermal image quality improvement

Figure 4.59: Distribution of Functions on Different Architectures

The set up is shown on the table. All setups will be using Jetson Nano as the primary platform. We will use AMG8833 for the thermal camera and IMX219-77 for the optical camera. The Jetson Nano's CPU will be used for running the main function of the thermal imager for all setups. The main function will be used for memory management, image stream decoding and encoding, and localization of temperature reading sites.

We will use CUDA Programming on Jetson Nano to separate which processes to run on the CPU and GPU. [94, 95] Experimental Group 1 will run face detection and image quality improvement on GPU, while Experimental Group 2 will run face detection and image quality improvement on a GPU-enabled server. These two processes will be run on the CPU, the GPU, and the server since these are the most computationally expensive processes.

4.6.2 Face Detection and Image Quality Improvement

For face detection, we will use convolutional neural networks (CNNs). The CNN will be trained using masked face data sets [24, 25] in order to be able to detect faces covered with face masks. A set of images taken using the optical camera prior to the initial testing will be used for timing the trained neural network on CPU, on GPU, and on the GPU-enabled server.

For image quality improvement, we will process a set of images taken using the thermal camera prior to the initial testing. We will use interpolation to improve the quality of the 8x8 thermal images. This does not change the temperature reading per pixel but this will be helpful for cross referencing the thermal images and optical images.

4.6.3 Initial data gathering and analysis

In order to demonstrate the speed-up, we will be comparing the latencies of processing computationally expensive tasks on CPU, on GPU, and on a GPU-enabled server. We will run and time face detection and image quality improvement on the setups. The data that will be collected will be the elapsed times per setup. The speedup will be computed by the eqaution:

$$\text{Speedup} = \frac{\text{Elapsed Time of Experimental Setup}}{\text{Elapsed Time of Control Group}}$$

The speedup is computed to show that the modifications to the thermal imager can increase the system's latency.

4.6.4 Processing Multiple Subjects In Real Time

In encoding and decoding the image stream from both the thermal camera and the optical camera, we will use Gstreamer, a builtin multimedia API of the Jetson Nano. The face detection algorithm will be modified in order to process multiple subjects in real time. The interpolation algorithm will also be modified in order to improve thermal images when faces are detected.

4.6.5 Localization of Temperature Reading Sites

Since the resolution of the images are not the same, we will align the optical images and improved thermal images by using field of view alignment. This will be done by transforming the thermal image based on the known angles of the optical camera and the thermal camera. The region of interest will be extracted by using the x and y coordinates of the detected face on the optical image and matching it with the thermal image. We will then measure the temperature corresponding to the localized area in order to take the temperature.

4.6.6 Final data gathering and analysis

All the processes will be integrated in order to create a thermal imager. The data that will be collected are the thermal imager's latency and throughput. Latency will be measured by timing the instance the image is fed into the CNN until the measured temperature is indicated. Throughput will be measured by the number of subjects successfully scanned and measured for 30 minutes.

Bibliography

[1] L. M. Ramon and S. H. Bello, *Dti and dole interim guidelines on workplace prevention and control of covid19*, Apr. 2020.

[2] C. for Devices and R. Health, *Thermal imaging (infrared thermographic systems / thermal cameras)*. [Online]. Available: https://www.fda.gov/medical-devices/general-hospital-devices-and-supplies/thermal-imaging-systems-infrared-thermographic-systems-thermal-imaging-cameras.

[3] *Advice for the public on covid-19*. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public.

[4] *Temperature guns versus thermal imaging technology*. [Online]. Available: https://www.flir.com/discover/rd-science/temperature-guns-versus-thermal-imaging-technology/.

[5] *10 best choices in 2021*. [Online]. Available: https://cheapestthermalcamera.com/.

[6] *International vocabulary of metrology*, *International Organization of Legal Metrology*. [Online]. Available: https://www.oiml.org/en/files/pdf_v/v002-200-e07.pdf (visited on 01/12/2021).

[7] E. Ring, A. Jung, B. Kalicki, J. Z. A. Rustecka, and R.Vardasca, *New standards for fever screening with thermal imaging systems*, 2013. [Online]. Available: https://doi.org/10.1142/S0219519413500450 (visited on 12/18/2020).

[8] T. Lewicki and K. Liu, *Ai thermometer for temperature screening: Demo abstract*, 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3384419.3430433 (visited on 12/04/2020).

[9] C. Hedge and Z. Jiang, *Autotriage - an open source edge computing raspberry pi ...* 2020. [Online]. Available: https://www.medrxiv.org/content/10.1101/2020.04.09.20059840v1.full.pdf (visited on 12/04/2020).

[10] *Clinical evaluation of fever-screening thermography: Impact of consensus guidelines and facial measurement location*, *Journal of Biomedical Optics*, 2020. [Online]. Available: https://www.spiedigitallibrary.org/journals/journal-of-biomedical-optics/volume-25/issue-09/097002/Clinical-evaluation-of-fever-screening-thermography--impact-of-consensus/10.1117/1.JBO.25.9.097002.full?SSO=1 (visited on 12/04/2020).

[11] ISO/CASCO, *General requirements for the competence of testing and calibration laboratories*, iso.org, Nov. 2017. [Online]. Available: https://www.iso.org/standard/66912.html (visited on 04/14/2021).

[12] MoviTherm, *Can i perform a thermal camera calibration myself?*, movitherm.com, 2018. [Online]. Available: https://movitherm.com/knowledgebase/thermal-camera-calibration/ (visited on 2018).

[13] *Grid-eye characteristics*, *Panasonic*. [Online]. Available: https://mediap.industry.panasonic.eu/assets/custom-upload/Components/Sensors/Industrial\%20Sensors/Infrared\%20Array\%20Sensor\%20Grid-EYE/application_notes_grid-eye_0.pdf (visited on 01/12/2021).

[14] *Why averages are often wrong*, *Flir.eu*, 2021. [Online]. Available: https://towardsdatascience.com/why-averages-are-often-wrong-1ff08e409a5b (visited on 01/19/2021).

[15] A. Coluccia, *A low-complexity approach for improving the accuracy of sensor networks*, *International Journal of Distributed Sensor Networks*, 2015. [Online]. Available: https://www.researchgate.net/

publication/282382252_A_Low-Complexity_Approach_for_Improving_the_Accuracy_of_Sensor_Networks (visited on 01/19/2021).

[16] E. Yusuf, H. Syamsudin, M. N. Mohammed, and S Al-Zubaidi, *2019 novel coronavirus disease (covid-19): Thermal imaging system for covid-19 symptom detection using iot technology*, 2020. [Online]. Available: https://www.revistaclinicapsicologica.com/data-cms/articles/20201021033943pmSSCI-271.pdf, [Accessed: 18-Dec-2020].

[17] T. Keary, *Latency vs throughput - understanding the difference & meaning*, Oct. 2020. [Online]. Available: https://www.comparitech.com/net-admin/latency-vs-throughput/, [Accessed: 30-Jan-2021].

[18] W. Wojcik, K. Gromaszek, and M. Junisbekov, *Face recognition: Issues, methods and alternative applications*, Jul. 2016. [Online]. Available: https://www.intechopen.com/books/face-recognition-semisupervised-classification-subspace-projection-and-evaluation-methods/face-recognition-issues-methods-and-alternative-applications, [Accessed: 30-Jan-2021].

[19] T. Lindner, D. Wyrwal, M. Bialek, and P. Nowak, *Face recognition system based on a single-board computer*, Sep. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9201668, [Accessed: 21-Oct-2020].

[20] W. Zhang, D. Zhao, L. Xu, Z. LI, W. Gong, and J. Zhou, *Distributed embedded deep learning based real-time video processing*, Feb. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7844524, [Accessed: 28-Oct-2020].

[21] *Non-contact infrared thermometers*, Apr. 2020. [Online]. Available: https://www.fda.gov/medical-devices/general-hospital-devices-and-supplies/non-contact-infrared-thermometers, [Accessed: 30-Jan-2021].

[22] *Non-contact temperature measurement devices: Considerations for use in port of entry screening activities*, Aug. 2014. [Online]. Available: https://stacks.cdc.gov/view/cdc/24857/cdc_24857_DS1.pdf?, [Accessed: 30-Jan-2021].

[23] *Thermal imaging (infrared thermographic systems / thermal cameras)*, Jan. 2021. [Online]. Available: https://www.fda.gov/medical-devices/general-hospital-devices-and-supplies/thermal-imaging-systems-infrared-thermographic-systems-thermal-imaging-cameras, [Accessed: 30-Jan-2021].

[24] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, *Maskedface-net - a dataset of correctly/incorrectly masked face images in the context of covid-19*, Nov. 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7837194/, [Accessed: 15-Mar-2021].

[25] X Zhangyang, *Real-world-masked-face-dataset*, Jan. 2021. [Online]. Available: https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset, [Accessed: 15-March-2021].

[26] A. B. Bondi, *Characteristics of scalability and their impact on performance*, Sep. 2000. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/350391.350432, [Accessed: 6-Jan-2021].

[27] R. Singh, S. Armour, A. Khan, M. Sooriyabandara, and G. Oikonomou, *The advantage of computation offloading in multi-access edge computing*, Jun. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8795335, [Accessed: 15-Feb-2021].

[28] *Flir screen est user manual*. [Online]. Available: https://scv-sa.ch/wp-content/uploads/2020/09/Users-manual-FLIR-EST-Thermal-Screening_09-2020.pdf (visited on 01/22/2021).

[29] *Ds-k1t671-3xf series face recognition terminal user manual*. [Online]. Available: https://mediap.industry.panasonic.eu/assets/imported/industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ADI8000C66.pdf (visited on 01/22/2021).

[30] *Manageability*, *Cambridge University*. [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/manageability (visited on 01/12/2021).

[31] M. Wilson, *Systems management - what is it and a full overview*, PC & Network Downloads - PCWDLD.com, Feb. 2021. [Online]. Available: https://www.pcwdld.com/systems-management (visited on 01/12/2021).

[32] T. Contributor, *Fcaps (fault-management, configuration, accounting, performance, and security)*, SearchNetworking, Apr. 2007. [Online]. Available: https://searchnetworking.techtarget.com/definition/FCAPS (visited on 01/12/2021).

[33] *Lpow forehead thermometer for adults, the non contact infrared baby thermometer for fever, body thermometer and surface thermometer 2 in 1 dual mode medical thermometer*. [Online]. Available: https://www.amazon.com/Forehead-Thermometer-Contact-Infrared-Surface/dp/B0861SFPM1.

[34] [Online]. Available: https://ihealthlabs.com/products/no-touch-infrared-forehead-thermometer.

[35] [Online]. Available: https://www.amazon.com/CandyCare-Thermometer-Measurement-Non-Contact-Temperature/dp/B086P4G3WR.

[36] E. Maras, *Temperature check kiosks ready to tackle covid-19*, QSRweb, May 2020. [Online]. Available: https://www.qsrweb.com/blogs/temperature-check-kiosks-ready-to-tackle-covid-19/ (visited on 01/12/2021).

[37] [Online]. Available: https://www.lamasatech.com/.

[38] [Online]. Available: https://intraedge.com/products/janus/.

[39] N. Nissim, R. Yahalom, and Y. Elovici, "Usb-based attacks", *Computers & Security*, vol. 70, Aug. 2017. DOI: 10.1016/j.cose.2017.08.002.

[40] K. Crawley, *Bluetooth security risks explained*, Jun. 2020. [Online]. Available: https://cybersecurity.att.com/blogs/security-essentials/bluetooth-security-risks-explained.

[41] *Most common wireless network attacks*, Apr. 2018. [Online]. Available: https://www.webtitan.com/blog/most-common-wireless-network-attacks/.

[42] L. Companies, *Covid-19 temperature screening kiosks & scanners*. [Online]. Available: https://www.loffler.com/temperature-screening-kiosks-scanners-covid-19.

[43] R. Angwin. (). "The most well-known and trusted thermal camera companies", [Online]. Available: https://thermalimagingcamerareviews.com/thermal-camera-manufacturers/. (accessed: 04.15.2021).

[44] (). "Flir company history", [Online]. Available: https://www.flir.eu/about/company-history/. (accessed: 04.15.2021).

[45] *Fluke warranties and care plans*. [Online]. Available: https://www.fluke.com/en-us/support/warranties.

[46] [Online]. Available: https://www.flir.com/support-center/warranty/instruments/2-10-thermal-camera-warranty-from-flir/#:~:text=Most\%20FLIR\%20products\%20are\%20covered,labor\%20coverage\%20on\%20the\%20camera.

[47] E. Villa, N. Arteaga-Marrero, and J. Ruiz-Alzola, "Performance assessment of low-cost thermal cameras for medical applications", *Sensors*, vol. 20, no. 5, p. 1321, Feb. 2020. DOI: 10.3390/s20051321. [Online]. Available: https://doi.org/10.3390/s20051321.

[48] *Thermometer gun - buy thermometer gun at best price in philippines: Www.lazada.com.ph*. [Online]. Available: https://www.lazada.com.ph/catalog/?_keyori=ss&clickTrackInfo=textId--5333330498349982838__abId--213203__pvid--159a2d5f-3278-4c85-95ae-bf604a3776dc&from=suggest_normal&page=1&q=thermometer+gun&sort=pricedesc&spm=a2o4l.home.search.5.239e359dkYpmz4&sugg=thermometer+gun_4_1..

[49] N. Vyas, *Are infrared thermometers accurate?*, Jun. 2020. [Online]. Available: https://health.clevelandclinic.org/are-infrared-thermometers-accurate/#:~:text=Research\%20has\%20shown\%20that\%2C\%20when,true\%20in\%20mass\%20temperature\%20screenings..

[50] D. Yaffe-bellany, *'thermometer guns' on coronavirus front lines are 'notoriously not accurate'*, Feb. 2020. [Online]. Available: https://www.nytimes.com/2020/02/14/business/coronavirus-temperature-sensor-guns.html.

[51] G. Prentice, *Temperature measurement accuracy guidelines*, Jul. 2015. [Online]. Available: https://www.piprocessinstrumentation.com/instrumentation/temperature-measurement/article/15562602/temperature-measurement-accuracy-guidelines#:~:text=

Temperature\%20measurements\%20can\%20be\%20categorized,Those\%20that\%20do\%20require\%20accuracy..

[52] M. Q. Ng and S. S. Teoh, "Development of a low-cost thermal camera for electrical condition monitoring", *Universal Journal of Electrical and Electronic Engineering*, vol. 6, no. 5A, 94â99, 2019. DOI: 10.13189/ujeee.2019.061511.

[53] T. C. Wilkes, L. R. Stanger, J. R. Willmott, T. D. Pering, A. J. S. McGonigle, and R. A. England, "The development of a low-cost, near infrared, high-temperature thermal imaging system and its application to the retrieval of accurate lava lake temperatures at masaya volcano, nicaragua", *Remote Sensing*, vol. 10, no. 3, 2018, ISSN: 2072-4292. DOI: 10.3390/rs10030450. [Online]. Available: https://www.mdpi.com/2072-4292/10/3/450.

[54] *How thermal imaging helps scientists study volcanic eruptions*, Oct. 2020. [Online]. Available: https://www.fluke.com/en-us/learn/blog/thermal-imaging/how-scientist-use-thermal-imaging-to-forecast-changes.

[55] N. E. Kallmyer, H. J. Shin, E. A. Brem, W. J. Israelsen, and N. F. Reuel, "Nesting box imager: Contact-free, real-time measurement of activity, surface body temperature, and respiratory rate applied to hibernating mouse models", *PLOS Biology*, vol. 17, no. 7, Jul. 2019. DOI: 10.1371/journal.pbio.3000406.

[56] J. GÃŒttler, C. Georgoulas, and T. Bock, "Contactless fever measurement based on thermal imagery analysis", in *2016 IEEE Sensors Applications Symposium (SAS)*, 2016, pp. 1–6. DOI: 10.1109/SAS.2016.7479837.

[57] U Jayalatsumi, A. F. Naaz, K. Sravani, A Anusha, and A. Vasavi, "A low cost thermal imaging system for medical diagnostic applications", *International Journal of Engineering & Technology*, vol. 7, no. 3.27, pp. 314–317, 2018, ISSN: 2227-524X. DOI: 10.14419/ijet.v7i3.27.17897. [Online]. Available: https://www.sciencepubco.com/index.php/ijet/article/view/17897.

[58] A. Somboonkaew, P. Prempree, S. Vuttivong, J. Wetcharungsri, S. Porntheeraphat, S. Chanhorm, P. Pongsoon, R. Amarit, Y. Intaravanne, K. Chaitavon, and S. Sumriddetchkajorn, "Mobile-platform for automatic fever screening system based on infrared forehead temperature", Jul. 2017, pp. 1–4. DOI: 10.1109/OECC.2017.8114910.

[59] A. B. Haripriya, K. Sunitha, and B. Mahima, "Development of low-cost thermal imaging system as a preliminary screening instrument", *Procedia Computer Science*, vol. 172, pp. 283–288, 2020, 9th World Engineering Education Forum (WEEF 2019) Proceedings : Disruptive Engineering Education for Sustainable Development, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.05.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920313685.

[60] *What is flask python*. [Online]. Available: https://pythonbasics.org/what-is-flask-python/.

[61] Mar. 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[62] C. Bernstein, K. Brush, and A. S. Gillis, *What is mqtt and how does it work?*, 2021. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport.

[63] *Last will and testament - mqtt essentials: Part 9*, Mar. 2015. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/.

[64] *11 best ping sweep tools and guide*, 2021. [Online]. Available: https://www.dnsstuff.com/ping-sweep.

[65] Sbtinstruments, *How to deal with lwt and disconnect reason â· issue 28 â· sbtinstruments/asyncio-mqtt*. [Online]. Available: https://github.com/sbtinstruments/asyncio-mqtt/issues/28.

[66] *Scheduling tasks with cron*. [Online]. Available: https://www.raspberrypi.org/documentation/linux/usage/cron.md.

[67] A. Kili, *How to impose high cpu load and stress test on linux using âstress-ngâ tool*, 2017. [Online]. Available: https://www.tecmint.com/linux-cpu-load-stress-test-with-stress-ng-tool/.

[68] *What is jquery?* [Online]. Available: https://jquery.com/.

[69] *Infrared array sensor grid-eye (amg88)*, *Panasonic*. [Online]. Available: https://mediap.industry.panasonic.eu/assets/imported/industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ADI8000C66.pdf (visited on 01/12/2021).

[70] *Grid-eye 2nd generation*, *Panasonic*. [Online]. Available: https://mediap.industry.panasonic.eu/assets/custom-upload/Components/Sensors/Industrial\%20Sensors/Infrared\%20Array\%20Sensor\%20Grid-EYE/pan170221_grid_eye_flyer_english_2017_downloadversion.pdf (visited on 01/12/2021).

[71] O. C. Carrasco, *Gaussian mixture models explained*, 2019. [Online]. Available: https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95 (visited on 06/11/2021).

[72] *Unsupervised machine learning: What is, algorithms, example*, *guru99*. [Online]. Available: https://www.guru99.com/unsupervised-machine-learning.html, (accessed: 04.15.2021).

[73] S. Saha. (Dec. 2018). "A comprehensive guide to convolutional neural networks - the eli5 way", [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. (accessed: 04.15.2021).

[74] E. Lutins, *Gaussian mixture models explained*, 2017. [Online]. Available: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f (visited on 06/11/2021).

[75] *Camera module*, *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/ (visited on 01/12/2021).

[76] A. Geitgey, *Face recognition*, 2018. [Online]. Available: https://github.com/ageitgey/face_recognition.

[77] *Dlib*. [Online]. Available: http://dlib.net/.

[78] J. Hrisko, *Thermal camera analysis with raspberry pi (amg8833)*, *Maker Portal*, 2021. [Online]. Available: https://makersportal.com/blog/thermal-camera-analysis-with-raspberry-pi-amg8833 (visited on 01/19/2021).

[79] *Adafruit amg8833 8x8 thermal camera sensor, python and circuitpython*, *Adafruit*, 2021. [Online]. Available: https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor/python-circuitpython (visited on 01/19/2021).

[80] *Infrared array sensor grid-eye*, *Panasonic*. [Online]. Available: https://mediap.industry.panasonic.eu/assets/custom-upload/Components/Sensors/Industrial\%20Sensors/Infrared\%20Array\%20Sensor\%20Grid-EYE/grid_eye_reference_specifications_160205.pdf (visited on 01/12/2021).

[81] *Calibrating your infrared thermometer with a properly made ice bath*, *ThermoWorks*, 2021. [Online]. Available: https://www.thermoworks.com/infrared_tips_icebath_to_calibrate_infrared (visited on 04/15/2021).

[82] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial networks*, 2014. eprint: arXiv:1406.2661.

[83] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, 2016. eprint: arXiv:1611.07004.

[84] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, 2017. eprint: arXiv:1703.10593.

[85] M. M. M. Fidali, *An inexpensive blackbody model*, 9th International Conference on Quantitative InfraRed Thermograph, Jun. 2008. [Online]. Available: https://www.ndt.net/article/qirt2008/14_01_04.pdf (visited on 2008).

[86] *Casia iris image database v4*. [Online]. Available: http://biometrics.idealtest.org/ (visited on 05/10/2021).

[87] S. Mishraa, A. Czajka, P. Liang, D. Z. Chen, and X. S. Hu, *Cc-net: Image complexity guided network compression for biomedical image segmentation*, Jul. 2019. [Online]. Available: https://arxiv.org/abs/1901.01578.

[88] T. B. Olaf Ronneberger Philipp Fischer, *U-net: Convolutional networks for biomedical image segmentation*, May 2015. [Online]. Available: https://arxiv.org/abs/1505.04597.

[89] Z. Fang and A. Czajka, *Open source iris recognition hardware and software with presentation attack detection*, Aug. 2020. [Online]. Available: https://arxiv.org/abs/2008.08220.

[90] *Open source iris recognition hardware and software with presentation attack detection*. [Online]. Available: https://github.com/aczajka/iris-recognition---pm-diseased-human-driven-bsif (visited on 05/10/2021).

[91] J. Daugman, *How iris recognition works*, Jan. 2004. [Online]. Available: https://arxiv.org/abs/1505.04597.

[92] A. Czajka, D. Moreira, K. W. Bowyer, and P. Flynn, *Domain-specific human-inspired binarized statistical image features for iris recognition*, Nov. 2018. [Online]. Available: https://arxiv.org/abs/1807.05248.

[93] *What is a container?* [Online]. Available: https://www.docker.com/resources/what-container.

[94] A. Kumar, *Introduction to cuda programming with jetson nano : Nvidia jetson*, Jan. 2020. [Online]. Available: https://maker.pro/nvidia-jetson/tutorial/introduction-to-cuda-programming-with-jetson-nano, [Accessed: 15-March-2021].

[95] M. Harris, *An even easier introduction to cuda*, Jan. 2017. [Online]. Available: https://developer.nvidia.com/blog/even-easier-introduction-cuda/, [Accessed: 15-March-2021].

Appendix A

Expandability Resource Usage

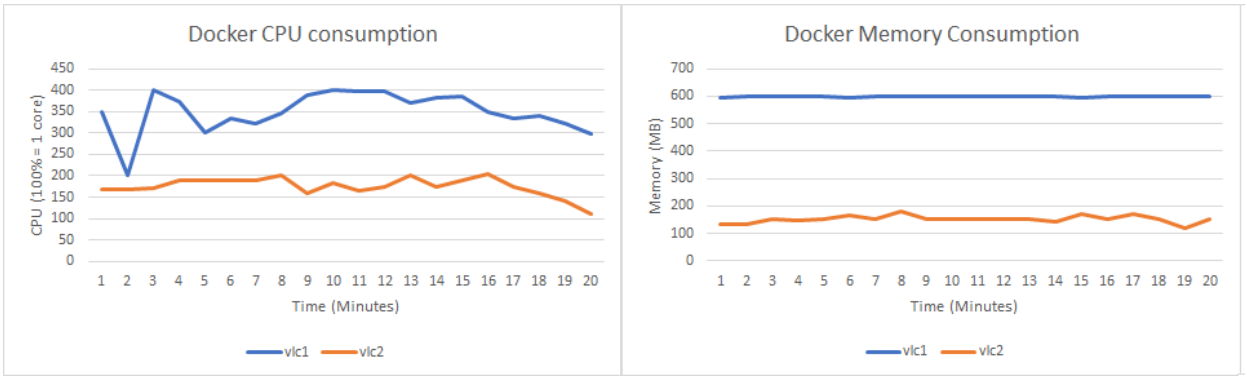


Figure A.1: Docker Consumption

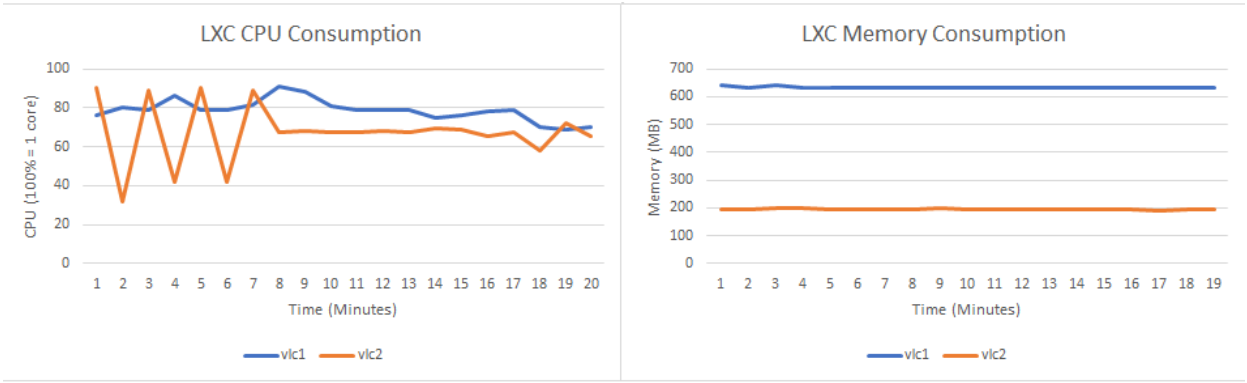


Figure A.2: LXC Consumption

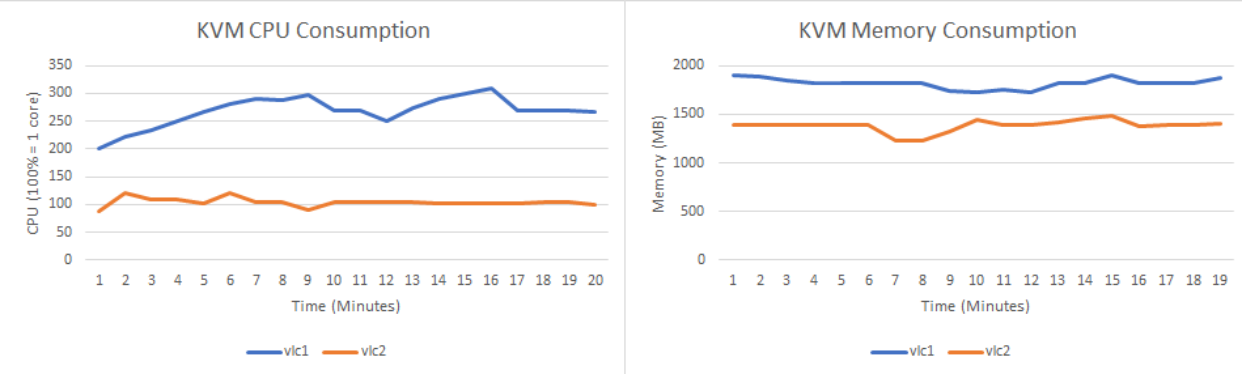


Figure A.3: KVM Consumption

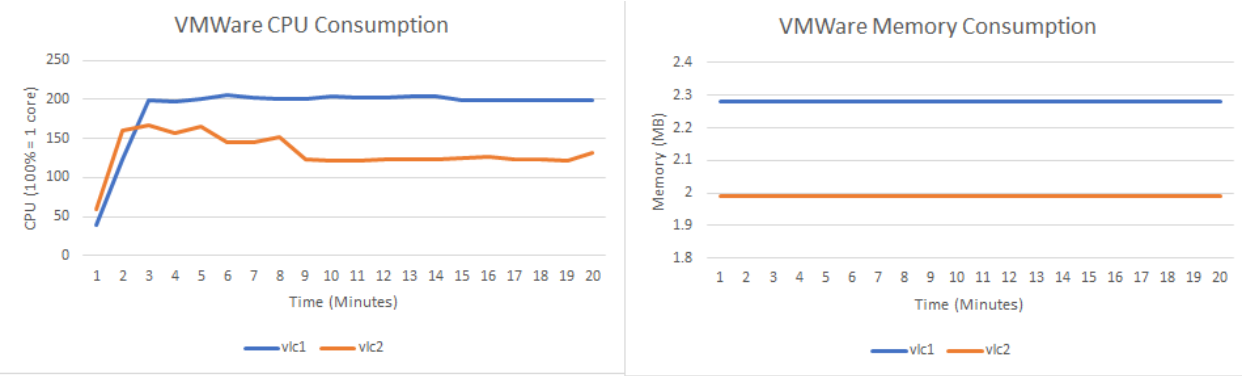


Figure A.4: VMWare Consumption