ESC470 Design Report

Team 1T6.5

Haoen Huang, 1000738570
Qi Ran Liao, 999848205
Xuan Tu, 1000740102
Dingjie Wang, 999876353

Table of Contents

1. Introduction

Thermal auditing is the process of detecting thermal leakages and evaluating thermal insulation of buildings such as residential houses, factories and public facilities. This process requires a capable technician to inspect the buildings with a thermal camera. Afterward, the technician will analyze the images, report any thermal leakages and suggest solutions to the customers. This capstone team saw the potential of automating this cumbersome process using advanced techniques in Machine Learning and Computer Vision. The team proposes a product that enables customers to perform thermal auditing themselves. It is our hope that this product paves the way for a more sustainable future.

Due to technical and time constraints, the problem has been scoped to thermal leakage detection for doors, with exploration into leakage detection for windows. Current practical and affordable thermal imaging camera resolutions are not high enough to allow diagnostics of an entire side of a house from a single picture (see section 2.2 for details). With the available thermal imaging capabilities, it is more practical to take pictures of particular objects of interest rather than entire sections of a house. Doors and windows were chosen due to being significant sources of thermal leakage, especially considering their relative surface area (see Appendix A). The project developers focused on heat loss during the winter due to

availability of data during the capstone period as well the higher average impact of heating vs. cooling on utilities bills (see Appendix A).

This document proposes an application that allows homeowners to evaluate the effectiveness of their home insulation. For maximum usability, this algorithm would ideally run standalone on a phone application. However, due to practical concerns (the immense amount of work required to reverse-engineer the logic required for communication with the seek camera and to work around dependencies in Java), we opted to try out two other approaches. We evaluate the trade-offs of two approaches: using a server to run our algorithm and using a raspberry pi to run our algorithm on device.

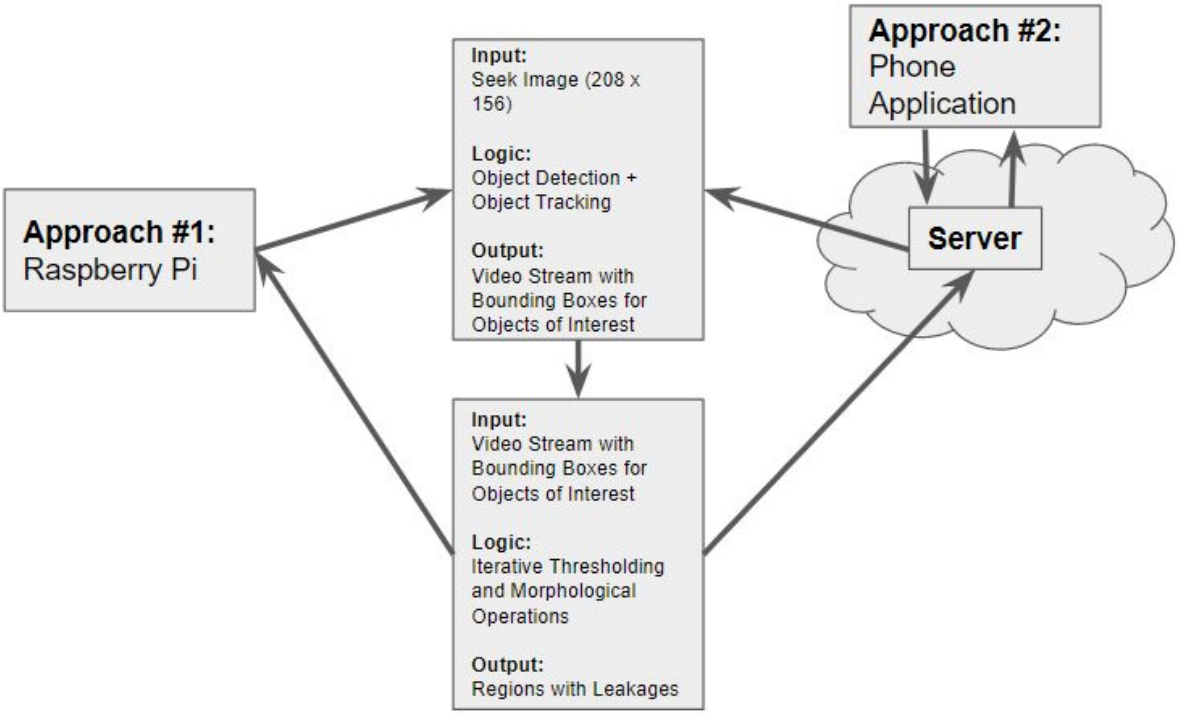


Figure 1. A diagram showing our two separate approaches. The 'Raspberry Pi' approach runs the algorithm standalone on device while the 'Phone Application' approach runs the algorithm on the cloud.

2. Key Design Decisions

2.1 Major Stakeholders

Note: The following stakeholders are ranked in order of relevance.

1. Capstone Project Developers

The capstone project developers are fourth year Engineering Science students from the Electrical and Computer Engineering option. Project developers are the primary stakeholders since they are entirely responsible for scoping, designing, and implementing automated thermal leakage detection. In addition to exploring a solution that can potentially reduce residential energy consumption and lower costs for homeowners, this stakeholder group seeks to develop technical skills that may benefit their upcoming work in industry. This stakeholder

group is also responsible for establishing a level of project scope and complexity appropriate for a fourth year design course.

2. Residents and homeowners
Residents and property owners represent the primary customer segment for automated thermal leakage detection. This stakeholder group can benefit from the project through reduced utility costs and a sense of contribution to improving the environment. Users of the product are interested in a reasonably priced product, user-friendly interface, accurate analysis of insulation problems, and practical solution recommendations.

3. Course instructor (Prof. Foster)
The course instructor manages the capstone design course and supports the project developers in the development process. This stakeholder group is interested in ensuring appropriate project scope and complexity, monitoring ongoing progress, and guiding students to make the most of their capstone experience. This stakeholder group expects a project that appropriately applies the technical and design skills acquired through the engineering science curriculum.

4. Department of Engineering Science
The Department of Engineering Science is interested in maintaining high standards of education and student engagement. This stakeholder group is also interested in ensuring appropriate project scope and complexity. Furthermore, this stakeholder group benefits from high quality projects that reflect and improve the reputation of the department.

5. Thermal Inspection Companies and Independent Home Inspectors
Thermal inspection companies (e.g. Great Northern Insulation) and independent home inspectors conduct commercial and residential thermal inspections. Trained technicians utilize thermal imaging for appraisal because it outperforms other diagnostic techniques in terms of accuracy, early detection, and non-invasiveness. While this stakeholder group may appear to be negatively impacted by the existence of a cheap/free automated solution, in reality they may stand to gain clients. Users of the automated thermal inspection solution may look to professionals for comprehensive insulation overhaul. This stakeholder group would benefit from a partnership that directs users to them for advanced insulation installation.

2.2 Camera Choice: Seek Thermal Compact for Android

2.2.1 Overview

The thermal camera is a critical component of our project. The solution's scale and feasibility are dependent on the capabilities of the selected camera. The camera's imaging specifications also determine the software design approach.

2.2.2 Objectives

1. Objectives for Project developers

    a. Maximize camera technical specifications to improve feasibility and fidelity of solution.
    b. Maximize industry-relevant learning in the given 3-month timeline.
    c. Maximize ease of interfacing with camera.
    d. Maximize ease with which camera can be integrated into an embedded system.
    e. Minimize cost of production.

2. Objectives for Homeowners
    a. Maximize ease of use for users without prior knowledge of thermal insulation.
    b. Minimize time required to generate results.
    c. Minimize cost of use.
    d. Maximize accuracy of solution recommendations.
    e. Maximize feasibility of applying recommended solutions.
    f. Maximize extensibility of solution with respect to future technological improvements.

3. Objectives for course instructors
    a. Demonstrate a level of complexity appropriate for a capstone design project.
    b. Demonstrate a level of rigor appropriate for a capstone design project.

4. Objectives for Department of Engineering Science
    a. Minimize cost to stay within budget.
    b. Maximize plausibility of future student use.

5. Objectives for Thermal Inspection Companies
    a. Direct application users to thermal inspection companies for complex solutions.

2.2.3 Metrics

Note: Metrics are listed for objectives that apply for the thermal camera comparison. Some objectives do not translate to meaningful metrics for thermal camera comparison.

1. Thermal imaging resolution (Objective 1a, 2d).
2. Thermal camera field of view (Objective 1a, 2d).
3. Thermal camera detection distance (Objective 1a, 2d).
4. Thermal imaging temperature range (Objective 1a, 2d).
5. Allowance for usage of popular and/or innovative software techniques (Objective 1b, 3a, 3b).
6. Ease of interfacing with camera.
7. Number of photos required to generate solution recommendations (Objective 2a, 2b).
8. Ease of use when taking photos (Objective 2a).
9. Camera cost (Objective 1e, Objective 2c, Objective 4a).
10. Camera portability (2a, 4a).

2.2.4 Overview of options considered

For lower cost and ease of software integration, the project developers chose among cameras that integrate with smartphones. The most popular entry-level thermal imaging cameras for smartphones are the Seek Thermal Compact, FLIR One, and FLIR One Pro. These cameras are popular for their price-to-performance ratio, phone compatibility, and proven reliability. Android and iPhone versions are available for all three cameras models.

2.2.5 Comparison Matrix

	Criteria	**Seek Thermal Compact**	**FLIR One**	**FLIR One Pro**
Imaging Resolution	Higher is better.	206 x 156. **Acceptable.** This resolution allows large objects such as the front door of a house to be assessed in a single image.	80 x 60. **Very poor.** This resolution does not allow large objects such as doors to be assessed in a single image, panning and multi-image capture is required.	160 x 120. **Poor.** This resolution may allow large objects such as doors to be assessed in a single image, but it must be close-up.
Field of View	Larger is better.	36 degrees. **Good.** This field of view allows image capture of objects at a reasonable viewing distance. Not a concern for the solution.	38 degrees. **Good.** This field of view allows image capture of objects at a reasonable viewing distance. Not a concern for the solution.	43 degrees. **Good.** This field of view allows image capture of objects at a reasonable viewing distance. Not a concern for the solution.
Detection Distance	Higher is better.	1000 feet listed. **Good.** This detection distance is sufficient for home inspection purposes.	At least 100 feet. **Good.** This detection distance is sufficient for home inspection purposes.	At least 100 feet. **Good.** This detection distance is sufficient for home inspection purposes.
Temperature Range	Larger is better.	-40 to 330 degrees celsius. **Good.** This temperature range is sufficient for home inspection in most regions globally.	-20 degrees to 120 degrees celsius. **Acceptable.** This temperature range is sufficient for home inspection in most regions globally, but it is not uncommon for temperatures to	-20 degrees to 400 degrees celsius. **Acceptable.** This temperature range is sufficient for home inspection in most regions globally, but it is not uncommon for temperatures to

			drop below -20 degrees celsius in some areas.	drop below -20 degrees celsius in some areas.
Usage of popular technologies	More is better.	**Good.** Allows application of machine learning and image processing techniques.	**Good.** Allows application of machine learning and image processing techniques.	**Good.** Allows application of machine learning and image processing techniques.
Ease of solution interfacing with camera	Higher is better.	No SDK released for development of third party applications. **Poor.** Difficult and time consuming to build a mobile app with native camera support.	SDK available for development of third party applications. **Good.** Easy to build a mobile app with native camera support.	SDK available for development of third party applications. **Good.** Easy to build a mobile app with native camera support.
Number of photos required for solution	Lower is better.	One image if close to object the size of a door. **Acceptable.** Taking a single picture is easy and intuitive.	Panning required to stitch multiple thermal images. **Poor.** Panning is inconvenient for users.	Some panning may be required, or more close-up images. **Poor.** Panning is inconvenient for users.
Ease of camera use	Higher is better.	Only thermal imaging available. **Acceptable.** Can at times be difficult to identify objects while taking pictures.	Optical image overlaid on top of thermal image. **Good.** Optical overlay allows for easier object identification.	Optical image overlaid on top of thermal image. **Good.** Optical overlay allows for easier object identification.
Cost	Lower is better.	$249. **Good.** Within the budget provided by EngSci department. Reasonably affordable with a cost comparable to entry-level smartphones and mid-range speakers.	$260. **Good.** Within the budget provided by EngSci department. Reasonably affordable with a cost comparable to entry-level smartphones and mid-range speakers.	$530. **Poor.** Outside of the budget provided by EngSci department. Rather expensive with a cost comparable to some flagship smartphones and game consoles.
Portability	Higher	**Good.** Easily fits in	**Good.** Easily fits in	**Good.** Easily fits in

	is better.	the palm of the hand.	the palm of the hand.	the palm of the hand.

2.2.6 Decision and Justification

The Seek Thermal Compact camera was selected for the project. Imaging resolution, number of photos required for solution (related to resolution), cost, ease of use, and ease of interfacing with camera were the main deciding metrics. Other metrics show low discrepancies between options, with all product ratings being either acceptable or higher.

The Seek Thermal Compact camera had a significantly higher thermal resolution than both FLIR cameras, and is available at the lowest cost. Its higher thermal resolution allows users to take a single image to generate solution recommendations, which is far more convenient than panning to take multiple images. The cost of the FLIR One Pro was prohibitive. The FLIR cameras include an optical image overlaid on top of a thermal image, which allows more straightforward identification of objects. However, it was determined that objects can also be identified from thermal images alone.

The FLIR cameras have support for third-party applications, which makes it possible to create a more polished final product that allows users to take photos and receive results in-app. With the Seek Thermal Camera, it is prohibitively difficult to create an application that interfaces directly with the camera, but an application could be designed that allows users to upload images taken through the official Seek Thermal application. It is also possible to take an embedded systems approach, interfacing with the Seek Camera through a Raspberry Pi. Additionally, Seek Thermal is still working on an SDK that should be made available for developers in the future.

Based on these considerations, the Seek Thermal Compact was selected as the best balance of solution accuracy, ease of use, and cost.

2.3 Hardware Integration

2.3.1 Overview

There are several ways to control the Seek Camera. While a smartphone provides the easiest access to the Seek Camera, it is not easily integrated into our pipeline with the creation of a third-party application. Hence, the team has explored another way to control the Seek Camera, which is to build a standalone hardware solution.

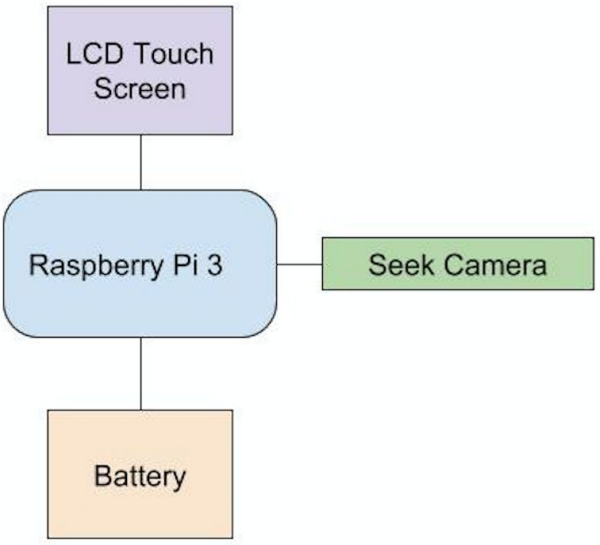


Fig.1 Schematic Diagram of Hardware Solution

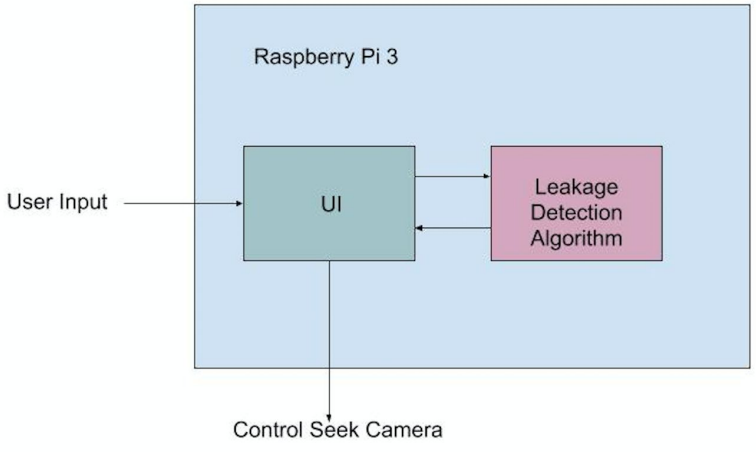


Fig.2 Software Module Deployed on Raspberry Pi 3

A Raspberry Pi 3 is used as the microcontroller. The Seek Camera is connected to the Raspberry Pi's Micro USB port. In order to control the Seek Camera, the Raspberry Pi runs a Python script which sends low level commands in byte stream to the Camera (think of it like

a driver). A 3.5’’ TFT LCD display is used to display output of the pipeline and debug messages.
The pseudo-code of python script:

```
def main():
    initialize_seek_camera()
    while(running):
        user_command = read_user_command()
        # Translate user-defined high level command to byte
      # stream understood by Seek Camera driver
        byte_stream = translateCommandToBytes(user_command)
        # Send the byte stream to the Seek Camera via USB
        status = sendBytesToCamera(byte_stream)
```

Here is the BOM of the system:

Material	**Quantity**	**Cost**
Raspberry Pi 3	1	$53
Wifi Dongle	1	$15
3.5’’ TFT LCD Display	1	$32
Seek Thermal Camera Compact Series	1	$249
Seek Thermal Camera Extension cord	1	$7

2.3.2 Objectives

1. Objectives for Project developers
    a. Maximize ease of development and environmental setup
    b. Maximize industry-relevant learning in the given 3-month timeline.
    c. Maximize ease of interfacing with camera.
2. Objectives for Homeowners
    d. Maximize ease of use for users without prior knowledge of thermal insulation.
    e. Minimize time required to generate results.
    f. Minimize cost of use.
    g. Maximize accuracy of solution recommendations.
    h. Maximize feasibility of applying recommended solutions.
3. Objectives for course instructors
    a. Demonstrate a level of complexity appropriate for a capstone design project.
    b. Demonstrate a level of rigor appropriate for a capstone design project.
4. Objectives for Department of Engineering Science

    a. Minimize cost to stay within budget.
    b. Maximize plausibility of future student use.
5. Objectives for Thermal Inspection Companies
    a. Direct application users to thermal inspection companies for complex solutions.

2.3.3 Metrics and Criteria

The following Metric and Criteria are developed based on the objectives wanted to achieved by various stakeholders.

1. Cost Of Manufacturing/Sale: lower the better
    - Link to objective 2f, 4a
2. Ease Of Camera Integration: higher the better
    - Link to objectives 1c, 3a
3. Compatibility With Other Programming Environment: more the better
    - Link to objectives 1a, 1b, 1c
4. Ease of Modification of Thermal Camera Image: higher the better
    - Link to objectives 1a, 5a, 2g, 2h
5. Computation Power of Microprocessor: higher the better
    - Link to objectives 1a, 1b, 2e
6. Ease of Use of By Homeowners: higher the better
    - Link to objectives 2d, 2e, 2g

	Raspberry Pi 3	Smartphone	BeagleBoard X15	iMX.6
Cost Of Manufacturing	**Good** It is the cheapest among these options. The average cost is $53	**Poor** Unless the user already has a phone. The average cost of a smartphone which is compatible with Seek Camera is $560	**Poor** The BeagleBoard X15 costs about $351	**Acceptable** The average cost of iMX.6 board with carrier is around $100
Ease Of Camera Integration	**Good** Plug and Play with driver written in Python. Access to raw image easily	**Acceptable** Plug and Play with the Seek Thermal app installed. Don't have access to raw image at all.	**Good** Plug and Play with driver written in Python. Access to raw image easily	**Good** Plug and Play with driver written in Python. Access to raw image easily

Compatibility With Other Programming Environment	**Good** Support a wide range of OS including many popular Linux Distro. Many third party libraries and language interpreters can be installed	**Poor** Only supports Java and the code is running on a customized JVM. Limited choice of third party libraries	**Good** Support a wide range of OS including many popular Linux Distro. Many third party libraries and language interpreters can be installed	**Good** Support a wide range of OS including many popular Linux Distro. Many third party libraries and language interpreters can be installed
Ease of Modification of Thermal Camera Image	**Good** Support Opencv and PIL	**Poor** Doesn't support Opencv well	**Good** Support Opencv and PIL	**Good** Support Opencv and PIL
Computation Power of Microprocessor	**Acceptable** Enough to drive LCD Display and thermal leakage detection code. Frame rate is not as smooth, ~ 2 frames per second. However, since it is used to take pictures not video, so it is not a huge problem	**Good** Enough to drive LCD Display and thermal leakage detection code. Frame rate is fast so the display is really smooth	**Good** Enough to drive LCD Display and thermal leakage detection code. Frame rate should be higher than Raspberry Pi 3 since its GPU is more powerful	**Good** Enough to drive LCD Display and thermal leakage detection code. Frame rate should be higher than Raspberry Pi 3 since its GPU is more powerful
Ease of Use of By Homeowners	**Poor** Homeowner needs to keep several extra pieces of hardware. I.E the board and a LCD display	**Good** No extra piece of hardware is required. Just a smartphone and camera. The UI also looks better on the phone's screen	**Poor** Homeowner needs to keep several extra pieces of hardware. I.E the board and a LCD display	**Poor** Homeowner needs to keep several extra pieces of hardware. I.E the board and a LCD display

2.3.4 Justification for using Raspberry Pi 3

The design decision of using a Raspberry Pi 3 is well justified given the criteria provided above. By using a Raspberry Pi as the microcontroller, the design team gains significant control of the Seek camera and gains the ability to develop customized APIs for integration with other software modules. For instance, the script developed above provides a driver-like

interface to the Seek camera and other software modules can easily access the camera for photo capture, camera parameter setup, etc. Furthermore, due to the nature of the Debian-based Linux kernel, gaining permission to communication ports such as the Micro USB is very simple. As a result, setting up the Seek Camera with a Raspberry Pi is simply plug-and-play. Furthermore, Raspberry Pi has GPIOs. As a result, instead of connecting to monitor, keyboard and mouse, it can be connected to an external LCD display or an external button. Raspberry Pi is known for its versatility as a low cost microcontroller. In this capstone project, this is beneficial since the team will need many external software modules installed on the microcontroller. The capstone team values the freedom of choosing the following aspects of a programming environment:

1. Operating System
2. Programming Language
3. Third-Party Support for Software Components

Therefore, these features enable the Raspberry Pi 3 to fit criterias 2 and 4 really well.

Raspberry Pi supports a large selection of operating system, such as Raspbian (a Debian Based Linux Distro optimized for Raspberry Pi), Ubuntu and other major Linux distros. The design team chose Raspbian as the primary OS for this capstone project. Furthermore, the design team requires the installation of Opencv and Tensorflow in order to implement other sections of the pipeline. All these software packages have been proven to install successfully on a Raspberry Pi running on Raspbian OS. Finally, the design team can use any major programming languages supported by a regular Linux-based OS. In this case, the design team chose Python, which makes software prototyping easy. This enables the Raspberry Pi 3 to fit criteria 3 really well.

Raspberry Pi is a very capable microcontroller with a powerful SoC (System on Chip, a single chip that contains important components such as CPU, GPU, RAM … etc) and shipped with various peripheral to interface with external devices. Even though it is not the most powerful microprocessor we considered, its cheaper cost makes the Raspberry Pi 3 well worth its value. Hence it fits criteria 1 really well and criteria 5 adequately.

2.3.5 Comparing Raspberry Pi 3 to a Smartphone

Raspberry Pi is a very capable microcontroller with a powerful SoC (System on Chip, a single chip that contains important components such as CPU, GPU, RAM … etc) and shipped with various peripheral to interface with external devices. Here is comparison of Raspberry Pi 3 computing power to a Samsung S8 smartphone (compatible with Seek Camera) :

	Raspberry Pi 3	**Samsung S8**
SoC	Broadcom BCM2837	Qualcomm MSM8998 Snapdragon 835

CPU	Quad-core 1.2 GHz 64-bit	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)
GPU	BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz	Qualcomm Adreno 540 710 MHz
RAM	SDRAM, 1GB	DRAM, 4GB
Permanent Storage	Vary, depends on size of MicroSD card.	64GB (internal) + up to 256GB external microSD

In general, the Samsung S8 outperforms Raspberry Pi 3 in all aspects. Even though doing a direct comparison between hardware specs doesn't necessarily represent the performance accurately, but according to the table, Samsung S8 has roughly twice as much computation power as Raspberry Pi 3. So a smartphone like Samsung S8 fits the criteria 5 well.

However, the trouble of getting our code run on the phone is very troublesome. There are issues with compatibility of USB OTG and the camera will only work with certain phones. Furthermore, the compatibility of other programming environment on a phone is terrible. It only allow Java and it has restricted choices of 3rd party library. Also, the Seek Camera Android app doesn't expose public APIs for developers to interact. Hence, using the combination of phone and camera fits the criteria 2, 3 ,4 very poorly. Nevertheless, if a homeowner has a compatible smartphone, then for them, using a phone fits criteria 6 better.

Nevertheless, Raspberry Pi 3 has sufficient computing power. Furthermore, the RAM of Raspberry Pi 3 is SDRAM which is twice as fast as DRAM. Furthermore, It has a reasonable GPU for its graphical processing. So it fits criteria 5 adequately, criteria 6 poorly and fits criteria 1, 2, 3, 4 much better than using a phone. Overall, the usage of Raspberry Pi 3 is preferred.

2.3.6 Comparing Raspberry Pi 3 To other Microprocessors

The team has also considered several other options for microprocessor. The team had looked into BeagleBoard X15 and Torodex Freescale i.MX 6. Some other microprocessors such as Arduino don't have enough processing power to drive a LCD touch screen and it doesn't have the capability to run an OS (needed by Python interpreter) so they are not considered at early stage of the design.

One strong candidate for the microprocessor is BeagleBoard X15. This board, supported by Texas instrument, is a very powerful microprocessor that outperforms Raspberry Pi 3 in terms of computing power. For instance, it has extra 1GB of RAM, a more powerful processor and extra ethernet port. It also has a mature community for all sorts of open source

projects. However, the cost of a BeagleBoard X15 is 4 times more ($239) than Raspberry Pi 3 and the extra power provided by BeagleBoard is not a necessity in this capstone. Therefore, it doesn't fit the objective of minimizing cost of manufacturing.

The last option is Torodex Freescale i.MX 6. One of the stakeholders (project developer) had experience using it before. So it is relatively easier to develop compared to two other solutions. At the same time, it is also more powerful than Raspberry Pi 3 in terms of computing power. It is capable of running OS and drive LCD touch screen. However, the cost and shipping from Switzerland will take a longer time.

Overall, both Beagleboard X15 and i.MX 6 out perform Raspberry Pi 3 in criteria 2 - 5 however, the cost of them is way higher than Raspberry Pi 3. So Raspberry Pi 3 fits criteria 1 much better than Beagleboard and i.MX 6 and preferred because Raspberry Pi 3 fits criteria 2 - 5 adequately.

2.3.7 Touch Screen LCD Display To Physical Button With LCD Display

3.5'' TFT LCD Display compared with a regular 3.5'' LCD Display with physical button is more user-friendly and the market of smartphone has proved that touch screen is better than keyboard. Therefore a TFT LCD Display fits criteria 6 better than regular LCD Display and it is used as the interface to homeowner. Despite it is more expensive than a regular LCD Display, the price of 3.5'' TFT LCD Display is $35 which is reasonable.

2.4 Object Detection

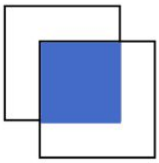
This stage of the pipeline takes grayscale thermal images as input, and outputs a list of bounding boxes where the objects of interests lie. The objects of interest are currently windows and doors but this will gradually expand as more and more objects are incorporated into the algorithm pipeline.

Design Requirements

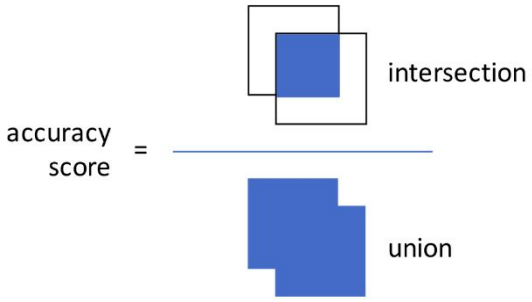
The stakeholders include the developer of this stage of the pipeline, the developer of the next stage of the pipeline, as well as the end-users. For the developer, the algorithms used will ideally be easily implementable (or be part of a popular library) with plenty of documentation. The next stage developer (leakage detector) will ideally obtain a perfect detection rate and a bounding box, as this will open up the possibility of better algorithms. Finally, the end-user would like to have fast runtime, and possibly even a version that runs in real time on their device.

We can summarize these requirements with a number of metrics which include: runtime, detection accuracy (preventing both false negatives and false positives), and bounding box accuracy. Detection accuracy can be measured by finding the inverse of the inaccuracy,

which is a combination of false negatives and false positives. Bounding box accuracy is slightly more difficult to quantify. There are a number of options for this:
We can calculate the overlap between the predicted region and the ground truth.



The issue with this is that a larger predicted bounding box that encompassed the ground truth would have 100%. We can improve on this by dividing the total by the union of the two sets.



Only a perfect match would have a 100% accuracy. The issue with this method is that it does not provide enough insight into the bounding box inaccuracies. When the bounding boxes for the ground truth were initially drawn, it was through a click-and-drag method. The top left corner would be selected by clicking on it, and the cursor was moved to the bottom right corner, where releasing the left click button would result in the selection of the bounding box. The issue with this was that the bottom right corner would be selected more accurately.

Note the image on the left. It demonstrates the possibility of error when annotating doors that are viewed from an angle using the click-and-drag method. The bottom right corner is often more accurate as the user can fix the alignment before releasing the mouse button. On the other hand, the click-and-drag method of annotation meant that the top-left corner, which was the initial click, loses out on accuracy. Notice that the left and top side are improperly aligned whereas the right and bottom corners enclosures are spot-on.
The breakdown of error would not show up if we used an overlap approach. Instead, the metric for bounding box accuracy was calculated to be the absolute percentage error of each of the four corners. This method revealed that the left and top sides of the bounding box were not as accurate (6%, 3%) when compared to the right and bottom sides (3%, 2%) and this may have been a result of the way in which annotations were done. The criteria for these metrics is lower runtime, higher accuracy, and higher bounding box

accuracy. Now that we fully understand the issue, metrics, and criteria, it is best to evaluate the current methods that exist.

An important distinction to note is that most applications of object detection deal with optical images (RGB) whereas the challenge presented to us is to directly use grayscale. The seek thermal camera output only shows the intensity values of the temperature. Fortunately, normalization has already been applied and the values are scaled to a grayscale image value between 0 to 255. An obvious obstacle to working with only thermal images is that we require differences in temperature in order to capture and differentiate different objects. This will be discussed as we explore each of the alternatives.

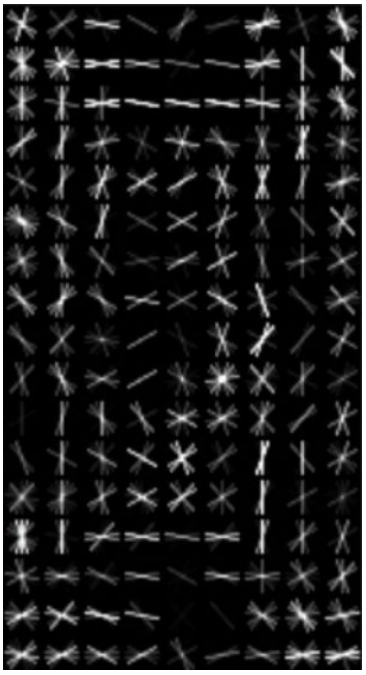


Figure 5. HOG mask.

Many object detection algorithms exist today. They can be broadly categorized as either a rigid model or a deep learning approach. The rigid model approaches include boosted classifiers (like the Viola Jones) and the histogram of oriented gradient. These models would be scaled and translated to check against different regions of the original image so it is able to achieve invariance to scale and translation. Different convolutional neural network models like the RCNN (Regional CNN) are used in the deep learning approach.

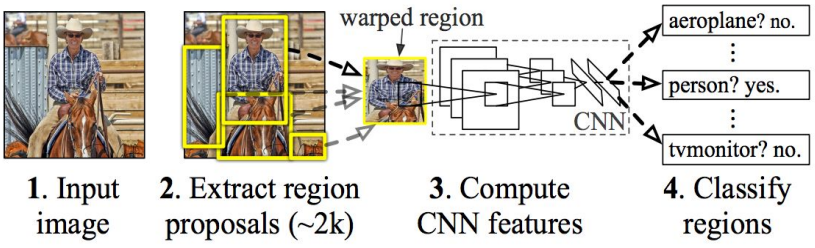


Figure 4. High Level Structure of RCNN.

There is a major trade-off to be made between using the rigid model approach vs. the deep learning approach: computational complexity and runtime. The rigid model approaches typically requires less computations whereas the CNN, although able to obtain state-of-the-art techniques accuracy, also has higher computational complexity and longer runtimes. Use of a CNN is often precluded by having a GPU.

Comparison of Different Approaches on Thermal Images of Doors		
Metric	**Convolutional Neural Network**	**Histogram of Oriented Gradients**
Runtime	30 FPS with GPU (Tesla K80) 30x longer w/o GPU	30+ FPS w/o GPU

Training Time (400 Images)	10 minutes (GPU) ~30x longer w/o GPU	2 minutes w/o GPU
Accuracy	98% Detection Rate	94% Detection Rate
Bounding Box Accuracy	Not measured. Inspected and seen as slightly less accurate. May be due to the lack of training data.	[6%, 3%, 3%, 2%]
Data Requirements	Smaller dataset is fine.	Requires large dataset for significant performance improvements.
Rotational Invariance	Rotational invariance can be easily obtained by augmenting dataset by adding rotated training images.	Does not scale as well. Multiple models are required, each with different rotation.
Aspect Ratio Invariance	Models are naturally invariant to aspect ratio because the proposal regions vary.	Does not scale well. Large differences in aspect ratio may demand more models.

The histogram of oriented gradients method was well-suited to doors because:

1. **Standardized aspect ratios.** Doors are very consistent in aspect ratios. The height of the door is on average six and a half feet while the width is roughly half the scale of the height. This benefits a rigid model approach because simple scaling and translations can generate the numerous possible combinations.
2. **Lack of rotation.** Users are unlikely to send an image that is extremely rotated where the edges of the door do not align with the x-axis or y-axis. The issue with a rigid model is that it does not check against rotation, but since this is not expected, the rigid model should perform fine.
3. **Singular object detection.** There is only one object that needs to be detected. If there are multiple objects, each must use its own separate model. The computational complexity scales linearly with the number of objects. A deep learning approach scales better with the number of objects as there will only need to be more connections in the outer (fully connection) layers.
4. **Properties of door.** The object itself can be aptly described with a rigid model. A typical door has leakage around the edges, which would be represented by a brighter region around the door. This brighter region would show up as a double gradient-edge which makes the histogram of oriented gradients an ideal approach.

It turns out that the rigid model approach is not as extensible of a solution. A standard aspect ratio is required. There are ways around this but it will lead to worse runtime. One example of a fix is to train a model with a re-scaling of its aspect ratio, and to then test the model on various images that are rescaled. In addition to increasing the runtime, it would lead to more

false positives. Therefore, we returned our focus to a deep learning approach. Training on optical has shown better recall and better accuracy. The reason for this appears to be the higher resolution images, and the fact that there is typically a larger contrast between walls and doors in optical images. As an improvement for the future, we may decide to use both optical and thermal images in our algorithm pipeline.

Deep Learning Framework

There are a number of open source deep learning libraries available. Four very popular choices appear to be Tensorflow, Caffe, Keras and PyTorch. The stakeholders are the developer for this stage of the pipeline, and the end user. The developer would benefit from having a easier learning curve. Furthermore, the framework needs to support the language that the developers have chosen to work in: Python. The popularity of these frameworks is very important as the ubiquity of the models determines whether there are existing models to choose from. The ones that were chosen for comparison are all relatively popular and have scripts to convert models between the various frameworks. For the end user, runtime is important. However, these models are likely to be run on a server since computational complexity is high. The runtime will be compared by running on a GPU.

Deep Learning Framework Comparison				
Framework	Tensorflow	Caffe	Keras	PyTorch
Python Support	Yes	Yes	Yes	Yes
Mobile Support	Good; Tensorflow Lite [6]	Great; Caffe2 [6]	Little or None	Little or None
Runtime*	Standard [5]	Fast	Slower [5]	Standard [5]
Learning Curve	Medium; Tensorflow has many abstractions that are not as intuitive: Sessions, tensors, etc.	Medium; There is less support for more complex networks as much of it is done in low level languages. Onus is on user to figure it out.	Low; High-level API makes building networks easy.	Low; A quick crawl through other user experiences on Quora indicates that PyTorch is easier to learn than Tensorflow, partly because of TF being described as unintuitive.

*Even though there are minor differences in runtime when tested on benchmarks, for our application it is not as important a distinction because even the slowest is acceptable.

The chart illustrates that many of these frameworks are comparable and any one of them could have been chosen. The most important factor is the ease of obtaining an existing RCNN model. Google has made available a new object detection framework, which was explored. Another implementation (Tensorbox) was also written in Tensorflow. Furthermore, the developers were already familiar with Tensorflow as it was taught in a course (ECE521) in the same semester. These practical concerns led us to choose to use Tensorflow for this project.

2.5 Leakage Detection and Classification

Given the location of the object, the next step is to detect potential leakages.The objective is to identify bright irregularities in the thermal image and determine if the region can be classified as a thermal leakage. The output will be outlines of leakage regions annotated with the type of leakage. From there, recommendations can be made to users to fix potential leakage issues. The primary focus of this project will be on doors specifically.

Detected areas of interest will be evaluated as either a false positive or a kind of leakage. For the simple case of doors, there are only two common issues that we are concerned with: heat loss through poor insulation and through air leakages.

The algorithm for leakage detection will rely heavily on image processing, specifically morphological transforms and thresholding techniques. Given the plethora of open source image processing libraries available to python, selecting an appropriate library to use was an important design decision.

Image processing library options considered:
OpenCV, SciPy, PIL, SimpleCV, with Numpy considered as a base comparison.

When deciding on the best library to use for image processing, several objectives concerning respective stakeholders were considered:

1. **Ease of use (project developers)** -- given the limited amount of experience developers have with image processing, ease of use was a very important criteria to consider.
2. **Functionality (project developers, homeowners)** -- a powerful library with more functions could improve the capabilities of the algorithm as well as help optimize the runtime.
3. **Compatibility (project developers)** -- since the code must be integrated onto a Raspberry Pi, the amount of dependencies is a major concern for project developers. Compatibility issues between packages can be very tedious to resolve.

The options are compared in the following table:

Objective	Metric	OpenCV	SciPy	PIL	SimpleCV
Ease of use	Quality and quantity of documentation	Very thorough	Acceptable	Good	Very thorough
Functionality	Amount of library functions	--	--	--	--
	Runtime	Very fast	fast	acceptable	acceptable
Compatibility	Amount of dependencies	Only Numpy and essentials like Cmake	Essentials, numpy, panda, few others	Zlib, libjpeg, openjpeg, etc	Numpy, OpenCV,Scipy,etc.

Values were determined from testing and reading documentation. The goal is to provide a relative measure of how each library compares with the others for the specified metric. The decision is not strictly based on any function of the scores; these valuations were provided to establish a comparison between the candidates within each objective.

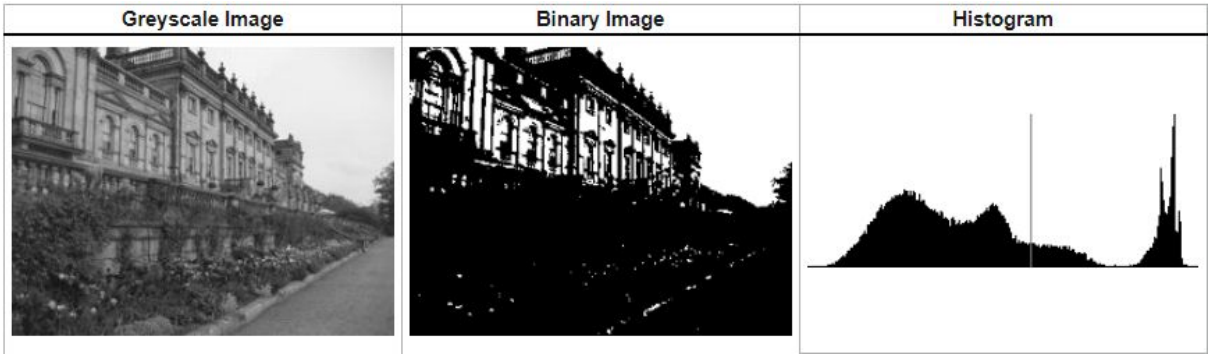
When comparing the amount of library functions, it was difficult to obtain this information for the SimpleCV and PIL libraries. Furthermore, the amount listed for SciPy includes functions dedicated to non-image processing purposes. Therefore, it was difficult to draw comparisons in this category.

OpenCV was selected mainly due to ease of use and the extensive resources provided for the library. Ease of use and compatibility were the main concerns. OpenCV provides detailed documentation specifying usage and examples of all library functions. Furthermore, the library also provides theoretical backgrounds and mathematical proofs on many functions and algorithms.

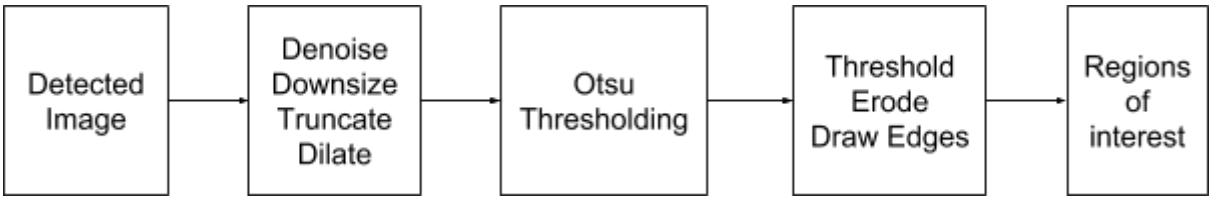
Functionality was not a major concern due to the relatively simple nature of the problem we are trying to solve; other libraries like SciPy and SimpleCV provide sufficient algorithms for our purposes. Fundamental image processing techniques are sufficient. However, when it comes to runtime, OpenCV was far superior to its competition, for example performing up to a magnitude faster than SciPy.

Algorithm Overview

The critical step in the leakage detection algorithm is Otsu Thresholding:



Thresholding at appropriate levels will conveniently identify regions of interest. First, the image is processed with a variety of filters to prepare for thresholding. A suitable threshold level is determined using the Otsu Thresholding Algorithm. The idea is to separate the pixels so that the spread within each class is a a minimum. As a result, identified regions will be concentrated bright spots. The modules are summarized in the following block diagram:

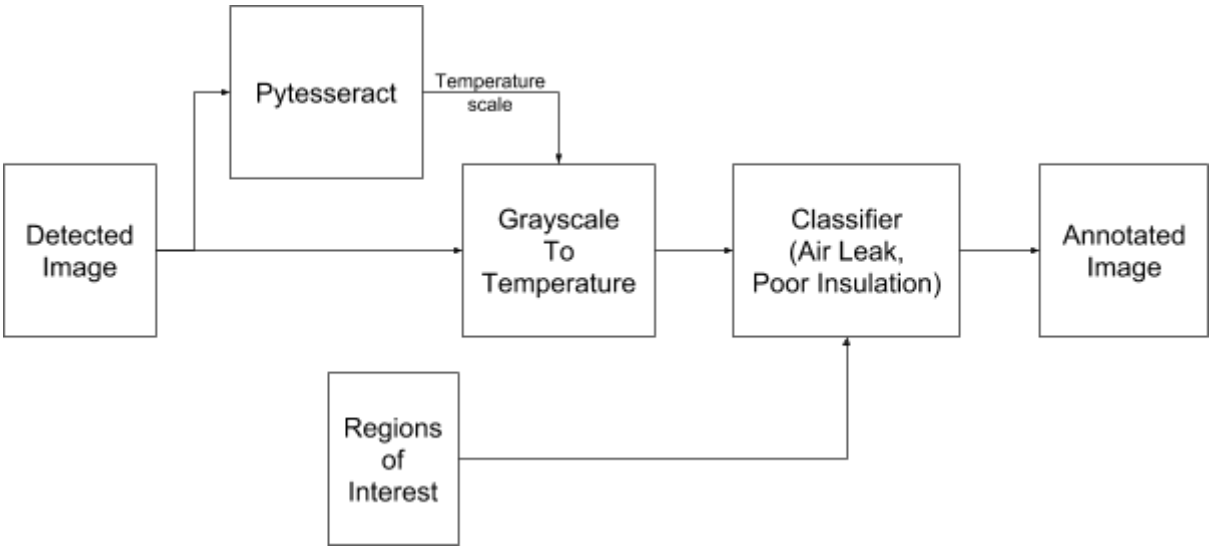


It is not possible to determine whether regions of interest actually represent leakages from the grayscale image alone, since the temperature scale is different for every image. Therefore, it is also necessary to translate the array of pixels into an array of temperature value. Once the relative temperature between regions are known, we can classify the types of leakages based on shape, location, and intensity of regions.

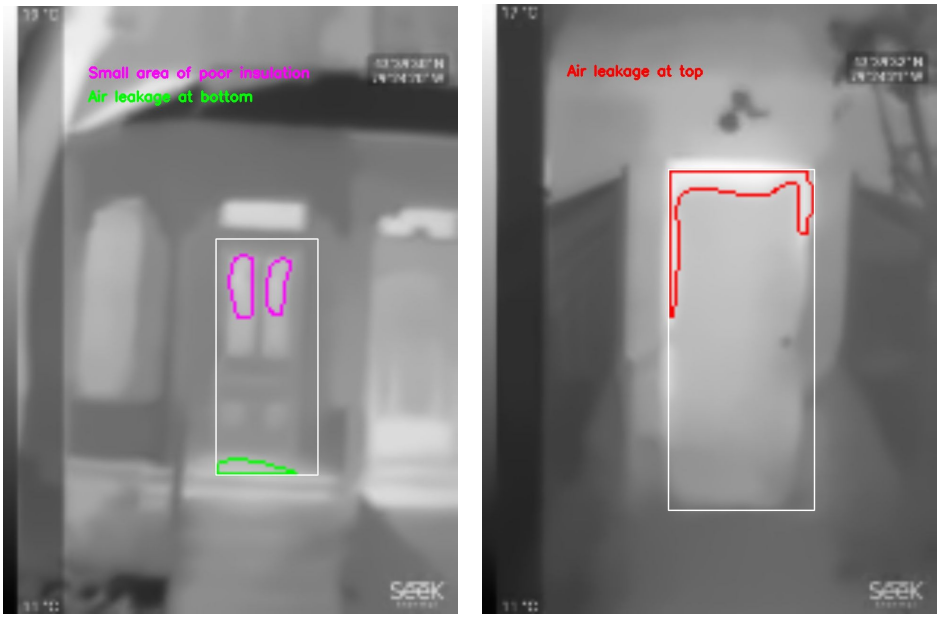
Once regions of interest are identified, they must be classified. For doors, we will assume that there will only two main types of leakages: poor insulation and air leaks. Poor insulation occurs when the door is made from materials with poor insulation properties, for example thin glass. Air leaks mainly result from a large split where the door meets the wall. Finally, a third possibility is that the region of interest is a false positive. Thresholding will always return a value and will therefore always identify a region as a local anomaly, however not every door has problems, and the filtered region may not actually represent a leakage.

The location of air leaks will also be specified because it affects the recommendations provided. For example, an air leak at the top or bottom of the door require different solutions than an air leak on the side with door hinges. Furthermore, the size of areas with poor insulation will be specified for the same reason. A large area of poor insulation may require placing a curtain behind the door or replacing the entire door; a small area may require more specific solutions to cover up a small portion of the door.

The modules used to classify and annotate regions of interest identified in the previous part are summarized below:



The end result is a labeled and classified image with annotations:



2.6 Backend Server choice

2.6.1 Overview

The backend server serves clients by accepting thermal images and returning labeled output images along with solution recommendations. The web client and backend server flow is presented as an alternative to the Raspberry Pi embedded solution. This solution demonstrates future extensibility, as a fully integrated Android application can be created once the Seek Thermal SDK for third-party applications is available. For the purposes of this capstone design project, the server is not required to support high scalability and availability, since it is only intended to demonstrate functionality.

2.6.2 Objectives

1. Objectives for Project developers
    a. Maximize ease of development.
    b. Maximize speed of development.
2. Objectives for Homeowners
    a. Minimize time required to generate results.

2.6.3 Overview of options considered

The choice of backend server does not have meaningful impact on most stakeholders. The main objectives were to easily integrate the server with existing processing code, minimize development time, and minimize time required to generate results. Python was chosen as the server language because all image processing is implemented in Python. Using the same language for the backend server allows integration of components into a single cohesive Python program with the server endpoints as the sole entry and exit points. Python Bottle, Flask, and Django were the three python web frameworks compared for this decision due to their popularity and community support.

Django is a full object-oriented and dynamically-typed web framework for building reusable web applications. It was released in 2005 and is considered a highly reliable and scalable framework. However, it is sometimes considered "heavy" due to its large overhead and number of dependencies. Flask is a microframework that has been rapidly adopted in industry. It was released in 2010 and was created on the concept on having a lightweight core framework that can be extended to include functionality as required. Bottle is a microframework that was designed to be fast, scalable, and lightweight. It is considered even "lighter" than bottle, with the philosophy of encapsulating all server functionality in a single file.

2.6.4 Decision and Justification

Since the choice of backend server does not have meaningful impact on most stakeholders at this stage, a full decision matrix analysis is not presented here. Instead, the decision is briefly described with respect to objectives outlined in section 2.6.2.

Bottle is the most lightweight framework of the three and is intended to be distributed as a single file module. The functionality of the server could be encapsulated within less than 150 lines, so single-file encapsulation suited our purposes well. Both Flask and Django have more setup overhead, requiring more code to provide basic server functionality. Bottle is very fast due to its lightweight nature, but all three web frameworks easily meet speed requirements as none of them would become a bottleneck compared to image processing time. Bottle was chosen as the web framework for its ease of use and speed of development.

2.7 Characterisation of air leakage through a pipe

2.7.1 Overview

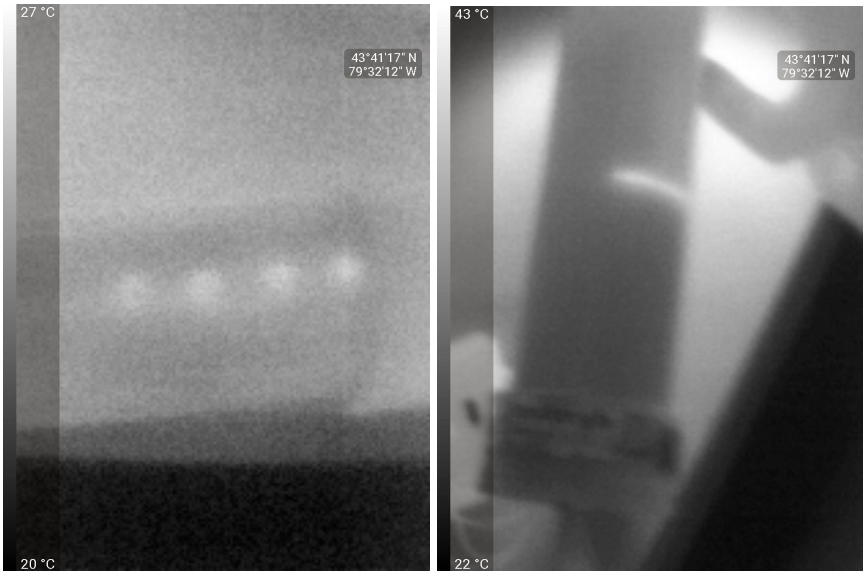
An exploratory experiment was conducted around the end of February in an attempt to characterize the properties of air leakage through a pipe. Project developers were interested in using thermal imaging to characterize plumes of air as they move between two environments with pressure and temperature differentials. The motivation was to explore the possibility of applying similar machine learning techniques to detect air /gas leakages and identify properties like pressure and temperature difference. This experiment will not be described in significant detail as it has already been demoed in earlier stages of the capstone, and its results do not impact final project results.

2.7.2 Experimental Setup

A PVC pipe was sealed with a rubber cap on one end with a pressure sensor and temperature sensors embedded inside the pipe. Air is blown in from the open end of the pipe, with air escaping through cracks created in interchangeable sections of PVC pipe. Various temperature and pressure differentials could be created by varying the air source (box fan,

blow dryer) and the ambient environment (room temperature, outdoor winter weather). The Seek thermal camera was used to capture images as the leak site for characterisation.

2.7.3 Experimental results



The pictures above show thermal images captured for air leakage through four holes and a horizontal slit, with a hair dryer as the heat source and ambient temperature at 20-22 degrees celsius. Unfortunately we found that while the thermal camera could detect heat at the leakage site, it lacked sufficient sensitivity to pick up actual plumes of air. Additionally, the image variation with respect to camera angle would make it prohibitively difficult to generalize results into a framework that can automatically characterize air leakage properties. For example, a picture of an air leak taken from the side would look entirely different from a picture that was taken from above or at a diagonal angle. While this is less of a concern for a home inspection application where users are asked to take pictures facing an object, it is an issue for applications such as gas detection where cameras remotely monitor pipes which can burst/split at any spot.

3. Challenges and solutions

3.1 Device Compatibility

The Seek Thermal Camera is only compatible with older models of phones that use micro-usb connection. Unfortunately, no member in the capstone group had a device that was compatible and outside help was required to find a compatible device.
While the device was compatible with Windows, it did not provide the portability to collect training data, nor is it a suitable device for a production application for leakage detection.

3.2 No Third Party Application Support

Seek Thermal™ does not provide any open-source software to work with its cameras. In comparison, FLIR™ provides a great deal of support for programmers to be able to create applications with their cameras. This made the option of creating a mobile application much

more difficult for this capstone project. We were forced to consider alternatives like using a remote server for processing, or using a raspberry pi.

3.3 Raspberry Pi challenges

Many challenges were encountered in setting the hardware and integrating thermal leakage detection code with the Raspberry Pi 3. A few highlights are presented below:

3.3.1 Compilation challenges

Compilation of required dependencies (noticeably, Opencv, Numpy, PIL) was challenging. Because those Python packages needed to be compiled from source and cross compiling was not set up. Therefore, to have complex packages compiled on a small memory (1GB), swap space of an extra 1GB needed to be setup on microSD card. Otherwise, there will be memory error during compilation stage.

3.3.2 Triggering code

The software package requires X Window System to be running (think of it as GUI) in the session. Hence, if the code is triggered via SSH, it will fail at an obscure location. To solve it, the code has to be triggered on the session which has the X Window System running.

3.3.3 Adjusting display size

Most of the relevant software packages (3rd party libraries) assume a display size of over 7''. However, the LCD Display the team used is 3.5''. This caused the UI display to overscan. The solution was to change the boot file to disable overscan. Even in this case the UI could still extend beyond the display. Therefore, the UI developed by the team had to be tuned and ported to such the smaller screen.

3.3.4 Python 2 vs Python 3

Some of the leakage detection code uses a higher version of Python and 3rd party libraries, so some part of the code did not immediately work on a Raspberry Pi 3 (Raspbian OS doesn't have Python 3). We opted to port the Python 3 code back to Python 2 compatible code.

3.3.5 Obtaining hardware components

It was challenging to acquire all required hardware to test the system. A reasonable amount of effort was put into purchasing a set of hardware for integration, with many components ordered online after failure to find them in hobby shops.

3.4 Google Cloud

A lot of effort went into setting up Google Cloud. The reason that a Google Cloud account was necessary was because I did not have the GPU resources to run training jobs locally. Requesting GPU resources itself took a day of correspondence because not everyone is allotted GPU resources when on a free-trial. There were some payment requirements.

Also, the process of setting up a Google Cloud took much longer than I anticipated. Unlike many other platforms that I later found out about (like Azure), Google Cloud did not provide scripts for building the GPU libraries on new instances. Much of the difficulty stemmed from inexperience of working with cloud instances (file transfer protocols and infrastructure specific commands), inexperience with GPU libraries and its dependencies. In total it took over a day and half to work out all the dependencies. Many of the errors are very obscure and difficult to debug, an example is "**CUDA_ERROR_UNKNOW**N", and often is a dependency issue.
A very important learning experience was when I realized that there was no 'save session' option for an instance. This became important when I realized the price of owning a GPU-enabled instance was 2-4 dollars per hour, which quickly adds up. I would essentially have to destroy my instance when I was not using it. I would have to re-install the dependencies all over. Each subsequent instance required less effort to re-setup as I automated much of the process. The culmination of all that effort was a startup script that can be found in the ***setup_scripts*** in the Github repository.

3.5 Tensorflow GPU Configuring

Tensorflow has a lot of issues that often arise. Here is a small sample of problems that required debugging:

- **Runtime appears to be the same as with CPU; CUDA_ERROR_NO_DEVICE:** The libraries are either incorrectly installed, or CUDA_VISIBLE_DEVICES environment flag needs to be set.
- **Could not allocate memory on GPU:** Tensorflow attempts to allocate all graph operations onto the GPU. Sometimes the graph is too large and this is impossible. A flag needs to be set so that this issue can be bypassed, and additional operations are performed by the CPU.

3.6 Leakage Analysis

Grayscale pixel values do not indicate temperature. The Seek thermal camera does not take pictures with fixed or preset temperature scales. Therefore, the difference in grayscale value between two regions does not have any correlation with the temperature; two areas vastly different in appearance can have small temperature differences if the scale is small.

The solution was to read the scale displayed on the side of the Seek thermal images and apply that information to convert pixel values to temperature values. This way, we can know the temperature difference between two regions. To read the scale, Pytesseract, a well known optical character recognition(OCR) library was used. Reading the scale was done in the preprocessing step to determine the pixel to temperature conversion formula.

Otsu Thresholding will filter out regions that are bright relative to its surroundings, but not necessarily compared to the rest of the picture. It's possible that the threshold returned

will clearly separate two regions that are relatively dark and do not reflect any thermal leakages. We are not interested in the variation of darker pixels as they cannot represent leakages.

This issue can be avoided by truncating the darker pixels that make up most of the picture and squashing them all into one value. This works based on the assumption that the leaks we are looking for are significantly warmer and brighter than the other parts of the door. As such, no valuable information is lost and threshold values that are too low will now be generated. Specifically, we can reasonably assume that the leakage values and the appropriate threshold value will be larger than the average grayscale value of the picture. Therefore, all values lower will be truncated to the average.

The best threshold for a leakage region only depends on the pixels close by, looking at the whole door is not necessary and can be misleading. The optimal threshold value that best outlines a region of interest will differ from region to region and is solely determined by the leakage area and its surroundings. Ideally, we can determine the best threshold value by performing Otsu's algorithm on the leakage area and a local region that wraps around it. However, to do this we need to know the answer already, the leakage regions.

Instead, we can guess where certain types of leaks should be and perform thresholding on those local regions. For example, a very common issue is air leakage through slits where the door meets the wall. Therefore, it is useful to perform thresholding on a region covering the edge of a door.

Differentiating between poor insulation and air leakages based on shape. We assume that air leakages almost only occur on the sides of the door. However, analyzing the location of a leakage region is not enough because it's possible that poor insulation can cover extend to the edges of doors.

Therefore, it is also helpful to analyze the shape of a leakage. Based on our assumption that air leakages mainly occur on the sides of doors, it is reasonable to assume that leakage regions will have a relatively slender shape. We can test how thin a shape is by assessing how much area is left after erosion.

3.7 Miscellaneous

The lack of large annotated leakage dataset precluded the use of convolutional neural network approaches for leakage classification. Furthermore, we cannot obtain many thermal images from indoors because this would require entry into individual homes.

Taking pictures was time consuming and exhausting. We took over 500 pictures of doors and windows, many of which were taken late at night to obtain the lowest outdoor

temperatures and reduce the impact of reflected sunlight on thermal sensing. It was slightly terrifying at times.

Annotation was a lot of work as well. We wrote our own scripts for creating annotations as many of the others do no interface well with each other (each model required a different annotation style) or were difficult to use (one custom script made it so that the process could not be paused in the middle of annotations and frequently crashed...).

4. Conclusion

Automated thermal leakage detection can potentially reduce cost for residents and homeowners while reducing energy consumption. The problem was framed to thermal leakage diagnostics for doors, with exploration into leakage diagnostics for windows. Decisions were made based on the priorities of relevant stakeholders, with a strong consideration for appropriate project scope and complexity. The final product allowed users to capture a thermal image of a door or window using a Seek Thermal camera, obtain thermal leakage diagnostics, and explore solution recommendations through links. Functionality was exposed via a server-based file-upload solution as well as a fully-integrated Raspberry Pi solution. The main obstacle for creating a fully-featured automated thermal insulation inspection solution was the difficulty of obtaining sample data. Two primary aspects of this challenge were a lack of access to varied house interiors and limited access to poorly insulated houses. The project developers feel that existing thermal inspection companies would be best equipped to develop a complete version of this application due to their abundant access to sample data.

References

[1] Anon, (n.d.). ***Thermal Cameras for your Smartphone*. [online] Available at: https://www.thermal.com/compact-series.html.**

[2] Anon, (n.d.). ***FLIR ONE Thermal Imaging Attachment for iOS and Android | FLIR Systems*. [online] Available at: http://www.flir.ca/flirone/ios-android/.**

[3] Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. ***2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).***

[4] Vieux, R., Benois-Pineau, J., Domenger, J. and Braquelaire, A. (2010). Segmentation-based multi-class semantic object detection. ***Multimedia Tools and Applications*, 60(2), pp.305-326.**

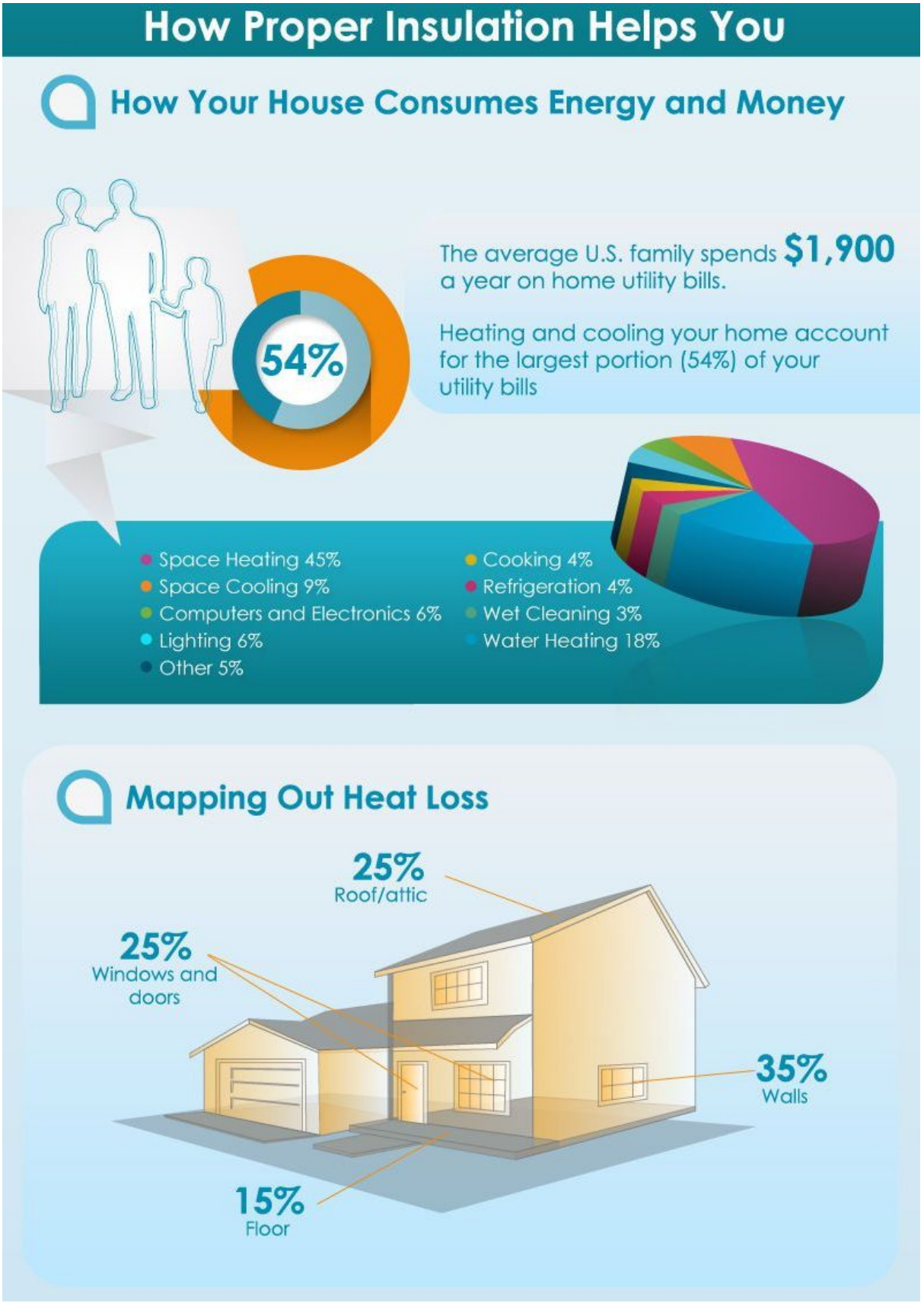
[5] Rosinski, W. (2018). *Deep Learning Frameworks Speed Comparison*. [online] **Deeply Thought. Available at: https://wrosinski.github.io/deep-learning-frameworks/ [Accessed 30 Apr. 2018].**

[6] Muhammad Zaheer. (2017). **How does caffe2 compare to tensorflow? [online] Quora. Available at: https://www.quora.com/How-does-Caffe-2-compare-to-TensorFlow**

[7] Dr. Andrew Greensted. (2010). ***Otsu Thresholding.* [online] Available at: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html**

[8] Green Home Gnome. (2018). ***Energy Loss in Homes and the Benefits of Insulation [infographic]*. [online] Available at: https://www.greenhomegnome.com/energy-loss-homes-insulation/ [Accessed 30 Apr. 2018].**

Appendix A: Thermal Insulation Infographic



Portion of home insulation infographic [6]