Early Forest Fire Detection System

by

Princepal Buttar

Arshdeep Gill

Senior Project

Electrical Engineering Department

California Polytechnic State University

San Luis Obispo

2014

Table of Contents

List of Figures

Acknowledgements

This project would not have been successful without the help and support from wonderful people and their valuable advice and feedback. First of all, we would like to thank Dr. Gary Hughes for all of his help. Dr. Hughes has been extremely helpful throughout this project; from giving us parts to build the system to helping us troubleshoot all the little issues. Without Dr. Hughes this project would not have been successful; he always took time out of his busy schedule to help us, even on holidays. Also our project advisors, Dr. John Jacobs and Dr. John Saghri, have always been there for us, providing valuable feedback on our weekly meetings. They kept pushing us so we would get this project done in time. Whenever we hit a roadblock they pointed us in the right direction. This project was made possible by the help of Raytheon; Dr. John Jacobs is from Raytheon and has collaborated with us on this project. So we would like to thank you for your time, support, and valuable feedback. Lastly we would like to thank David Garges, a master's student, for his input and helping us with various phases of this project.

Abstract

This project aims at building a prototype of a land-based early forest fire detection system using a long-wave infrared camera, powered by a 12 V motorcycle battery. It uses a system engineering approach by putting different software and hardware blocks together to come up with a unique *automated system* that detects fire in the field of view. The fire can be detected, from a distance of 232 feet if the fire is from a small stove or 64 feet if the fire is from a candle about the size of a quarter, within a second and sends a warning email as an alert to a specified email address. This project uses a simple fire detection Matlab algorithm based on thresholding to detect forest fire. The system is designed in a way that in future complicated detection algorithms can easily be integrated with the system.

Introduction

Forest fires are an immense burden to the environment, infrastructure, economy, and most importantly, human life around the world. In 2012, the United States spent an excess of $1.2 billion dollars trying to suppress fires that claimed over 9.3 million acres of land, 2,216 residences, 1,961 outbuildings, and 67 commercial structures. Early detection of forest fire can drastically reduce the loss. This senior project is an effort to build an automated real time early warning system as a measure to prevent forest fires, through detecting early signs of fire using a long range infrared camera. Since this project is more focused on building a fully functional demo of the system that could use any fire detection algorithms rather than focusing on just building a fire detecting algorithm as done in the past, a simple thresholding technique is being applied to the imagery data from the camera to detect fire in the scene within a second. Finally, an automated warning email is sent out by the system to any specified email address as an alert.

Requirements and Specifications

TABLE I

EARLY FIRE DETECTION REQUIREMENTS AND SPECIFICATIONS

Marketing Requirements	Engineering Specifications	Justification
1, 2, 4	Automatic day and night monitoring	Once initiated, the system should operate automatically anytime of the day
5	Detect fire within a few seconds	Detect a small fire fast and send a warning early enough to alert the Fire department right away
6	Detect fire up to 200 Feet	Long-wave infrared camera allows for detection of fire up to 200 feet for a medium size fire
2	Automatic Warning alert sent at any distance	Automatic email allows for easy and cost effective way for sending a warning message all over the globe
3	Unit powered by 12V battery	Unit will be mounted on remote locations therefore it should provide its own power
Marketing Requirements 1. 24/7 Fire monitoring system 2. Cost effective 3. Portable 4. Minimum human interaction 5. Fast processing 6. Long range		

Table I summarizes the marketing requirements and the Engineering Specifications that go with them.

System Architecture

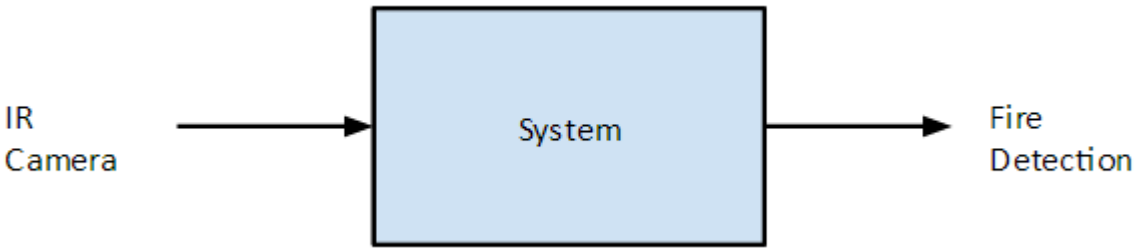


Figure 1: Zero-Level System Diagram

Figure 1 shows a zero-level block diagram of the system; infrared (IR) camera provides an input to the system which then processes the data from the camera and sends a warning signal if fire is detected.

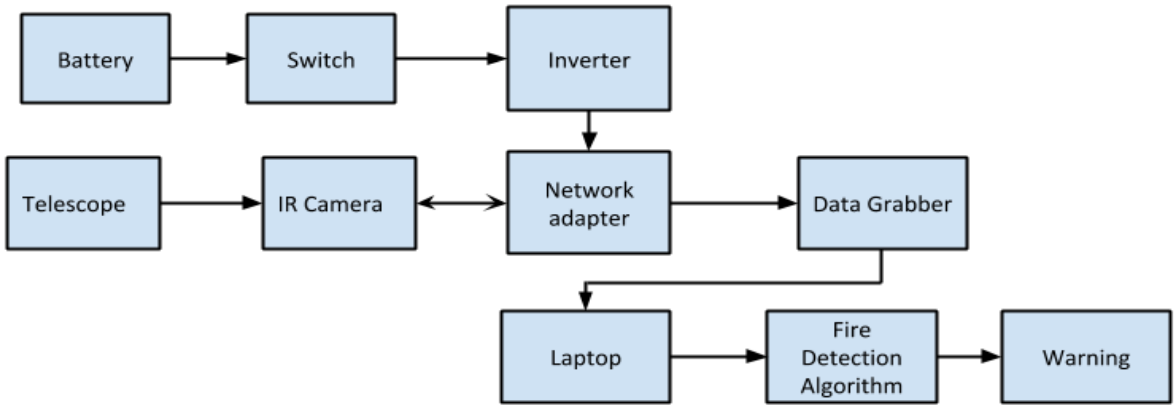


Figure 2: Level 1 block diagram

Figure 2 shows the level 1 diagram of the system. The long wave infrared (IR) camera is mounted on a telescope (*during the last testing phase*); this IR camera connects to a network adapter which connects to a laptop through a data grabber program, written in C++. This data is then processed using an algorithm in Matlab which sends an email as warning when fire is detected. The whole system is powered with a 12V motorcycle battery that connects to an

inverter through a switch. This inverter converts 12V DC to 110V AC which powers the network adapter and the IR camera. The laptop runs on its own battery.

Figure 3 shows the simplified flow chart of program that connects the laptop to the infrared camera. Basic structure of the program goes as following: all the needed variables are declared, then the program controlling the operation of the camera is setup to collect data and store it in a *.fits* file otherwise the program exits.

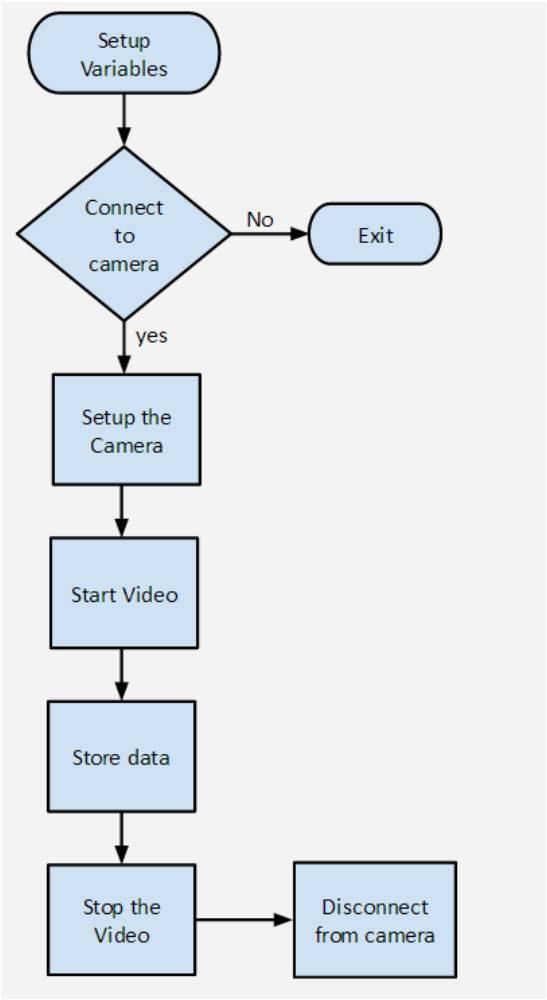


Figure 3: Simplified Flow Chart for program that connects to Camera

Component design:

HARDWARE

Long-wave Infrared Camera: In this project the FLIR Photon 640 camera with 35mm lens and f/1.4 is being used to take data. This camera takes a picture with a resolution of 512 X 640. The infrared camera needs to be able to detect fire, after some research and talking with our advisors; we decided that long-wave infrared camera meets the requirement. Long wave infrared region of the EM spectrum is sometimes also called the “thermal imaging” region. The infrared camera is connected to the network adapter with a VGA to GPIB cable. The VGA end

connects to the camera and GPIB end connects to the network Adapter. The VGA cable provides the needed power to the camera, therefore there are no other wires connected to the camera.

Network Adapter: The network Adapter connects between the camera and the computer. The digital data from the camera is sent to the computer through an Ethernet cable. But in order for the computer and the network adapter to communicate the IP address on the computer must be set, to be static (usually 192.168.2.1). Coyote, software provided by FLIR, must be used to make sure that the computer is connected to the camera. Also the network adapter provides power to the camera; and this adapter is powered by an AC to DC converter which is connected to a 12V battery.

Power Adapter: This converts the 110V AC to 9V DC at 1.66A to be used by the Network Adapter. This adapter looks like a regular laptop power adapter but with less power capabilities.

Battery: This is a 12V motorcycle battery which powers the network adapter and the camera. The battery is connected to the DC to AC converter through a switch. This battery is enough to last 4-5 hours on a full charge.

DC-AC converter: This is 150W DC-AC converter which is more than enough power to be able to handle the load of the power adapter, which theoretically only requires little over 15W. This DC-AC converter connects between the battery and the network adapter. It converts the 12V DC to 110V AC, so that the power can be used by the network adapter.

Laptop: Laptop saves all the data and process the data to figure out if there is fire in the scene. Laptop connects to the network adapter through an Ethernet cable. The laptop and the network must be setup to be able to communicate with each other, as explained in network adapter section. Laptop has Windows 7 64-bit, photon SDK, Matlab, and Microsoft Visual Studio to run the system. Photon SDK is needed to be able to communicate with the camera through the Ethernet cable. Matlab automates the process of grabbing the data, processing each frame, and sending a warning email if fire is detected. Microsoft Visual Studio is only needed if C++ software needs to be recompiled and updated executable is needed. The current executable works flawlessly with the current setup, so there should be no need for Microsoft Visual Studio unless the setup changes. With the current setup the laptop lasts about 3 hours.

SOFTWARE

C++ Code:

In order to compile and run the program the computer must have the Photon SDK installed. Figure 4 shows the detailed flow chart of the program that connects to the infrared camera. All of the variables are setup, and then the program tries to connect to the camera. If it cannot connect to the camera, then the program exits. If the laptop can correctly communicate with

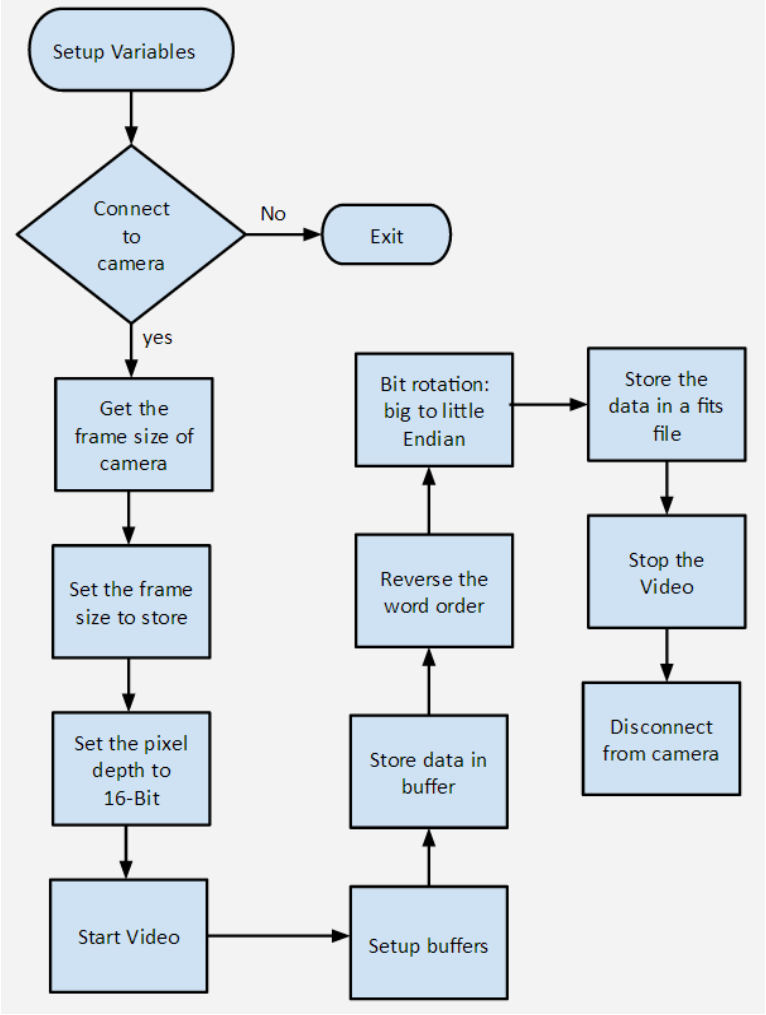


Figure 4: Detailed Flow Chart of program that connects to camera

the camera then the camera is setup to store the data. First step in setting up the camera is to determine the frame size and then that frame size is used to store data. Also the pixel depth of the camera is set to 16-bit. Then the video is started to allow the computer to grab data from the computer. Three different buffers are setup as following: the first buffer stores the raw data from the camera, then the word order is reversed and stored in the second buffer, and lastly bits are rotated to convert data from BigEndian to LittleEndian. This is done so that the data can be stored in fits image file. Finally the video is stopped so that program can properly disconnect from the camera. This program also writes the header of the fits file, which an important part of the fits image file. The header describes the data stored in the file since the data stored in fits file doesn't necessary have to be an image data, it could be tabular data. Therefore the header describes the type of data and how it is stored.

MATLAB Code:

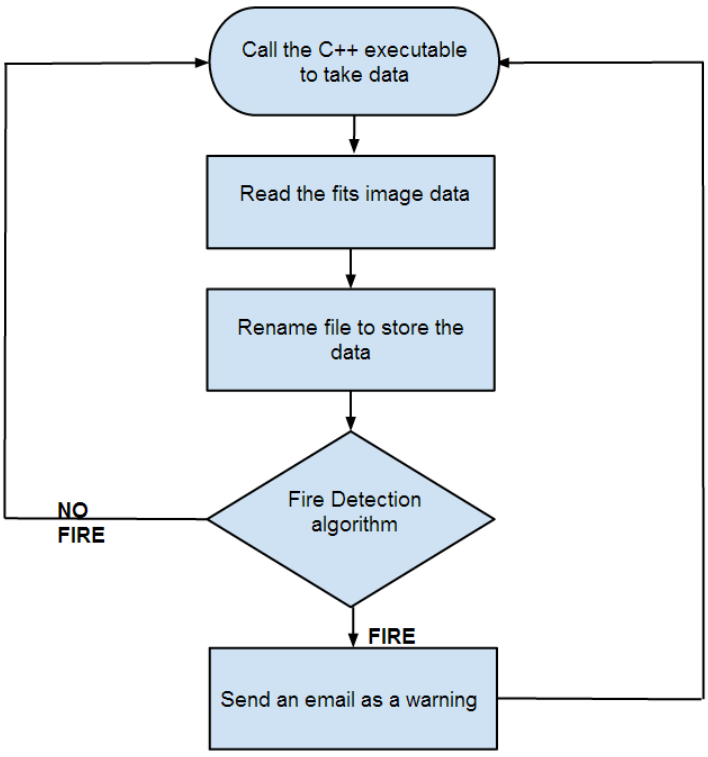


Figure 5: Matlab code flowchart

Figure 5 shows the functional flowchart for the Matlab code. The code once initiated runs in a while loop making the system automated. It starts with calling the C++ .exe file saved in the project folder on the PC. The executable as described in the previous section, helps in grabbing a raw 14-bit data frame

and save in the same directory in .fits format. As, the picture taken by the camera is flipped, every frame is rotated by 180 degrees and is overwritten with its rotated version. The C++ program saves every frame under the same name (*IRCAM.fits*), so the Matlab renames each frame to not get overwritten by the next frame in order to save all the data. The data is saved in sequential order with the names ending in numerical numbers starting with 1. For example, frame 1, frame2, frame3 and so on are saved as IRCAM1.fits, IRCAM2.fits, IRCAM3.fits, and so on respectively. The next step is the actual fire detection through thresholding, by choosing a pixel value corresponding to fire heat intensity through prior testing and then setting everything below that pixel value to zero. Anything above the chosen pixel value would correspond to heat levels of the actual fire. If such pixel values are present the code would detect fire and send an automatic warning email to any email address specified in the code. The email address specified in the code is a dedicated account for this project. Regardless of fire being detected or not, the code runs again to perform all the steps. The processing time for each frame is one second, this includes taking the picture, saving the data, and detecting the fire. The code is as shown in Appendix B.

Testing

Range testing experiments

To test the range capability of the system, three independent experiments using fire of three different intensities were conducted around 3 P.M. during a hot summer day with a day temperature of 92 □F. The three different fire sources included a candle, a stove, and burning newspaper as shown in the images below:

Figure 8: Candle with quarter for size comparison

Figure 7: Stove Fire

Figure 6: Newspaper Fire

NOTE: *All the highlighted values in the frames below are the maximum pixel values in the whole image and not just the maximum for the small pointer area. (Maximum values obtained using Matlab imtool first).*

The three experiments are described below:

Candle test

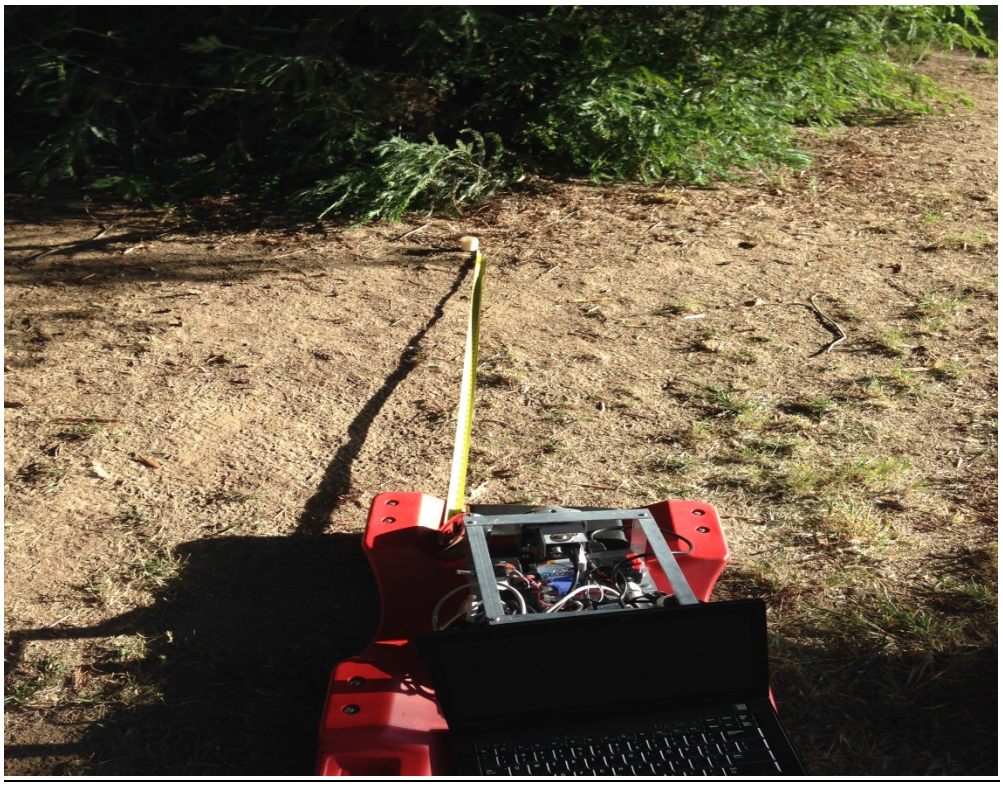
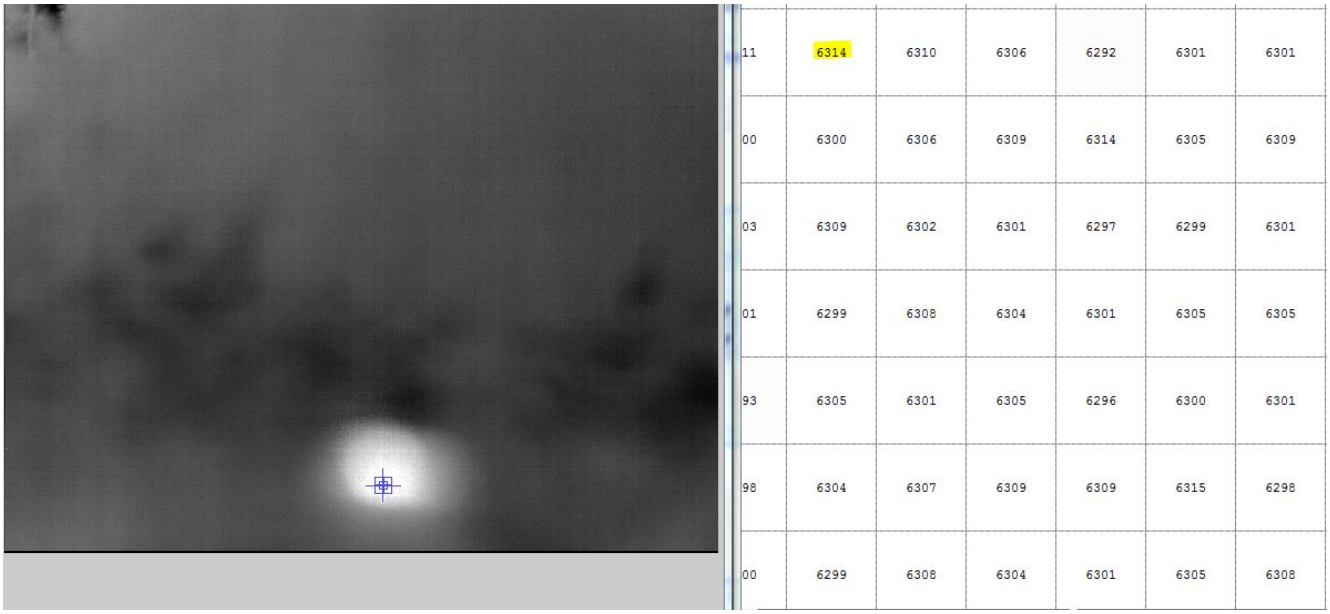


Figure 9: Experimental setup using a candle

Figure 9 shows the experimental setup for the system range testing using a candle, placed initially 5 feet away from the camera. A yellow measuring tape as seen in the figure was used to determine the distance between the system and the fire. A trolley was used as seen in the picture to vary the distance of the fire from the system easily.



11	6314	6310	6306	6292	6301	6301
00	6300	6306	6309	6314	6305	6309
03	6309	6302	6301	6297	6299	6301
01	6299	6308	6304	6301	6305	6305
93	6305	6301	6305	6296	6300	6301
98	6304	6307	6309	6309	6315	6298
00	6299	6308	6304	6301	6305	6308

Figure 10: Infrared image of the candle (5 ft. away) with pixel value chart (Max highlighted)

Figure 10 shows one of the infrared images taken by the camera during the experiment, with the pixel value chart on the side corresponding to the pointer area in the image, aimed at the candle flame. As seen from the table, the maximum pixel value of the flame is highlighted in yellow which reads 6314. This maximum value is way below the threshold (6800) for detecting fire, hence no fire was detected. Also it is noted that the image is not very focused, which is discussed after the next setup.

The candle test was also performed at another day around 4 P.M. with a day temperature of 90 □F. Initially the distance between the camera and the candle was 8 feet this time.

6	6148	6453	6626	6442	6537	6698
0	6205	6695	6668	6474	6462	7018
0	6202	6743	6749	6460	6496	6899
4	6150	6520	6764	6600	6578	6575
3	6091	6271	6462	6479	6420	6318
5	6072	6141	6233	6236	6209	6189

Figure 11: Infrared image of the candle (8 ft. away) with pixel value chart (Max highlighted)

Figure 11 shows the maximum pixel value of the flame from the same candle to be 7018, which is above the 6800 threshold value, so the fire was successfully detected this time.

In the second experiment since the distance was always more than about 8 Feet, the camera was able to focus on the small flame as oppose to the first experiment, where the distance was only 5 feet so the camera couldn't focus on the small flame and the fire wasn't detected. The minimum distance where the camera focuses automatically is 8 Feet.

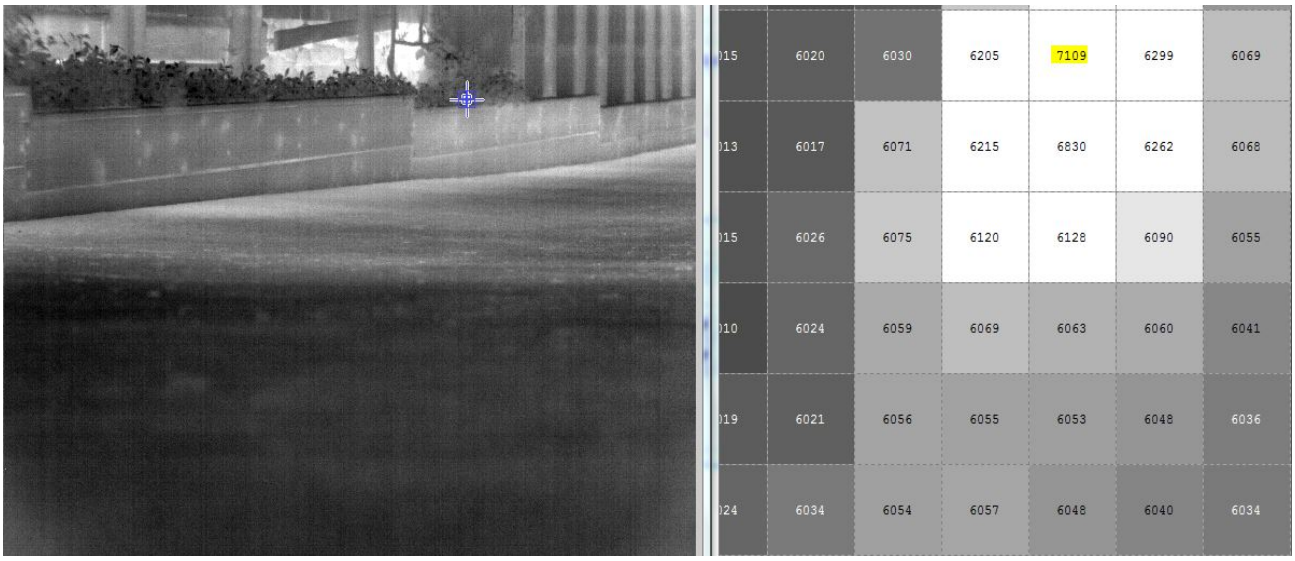


Figure 12: Infrared image of the candle (64 ft. away) with pixel value chart (Max highlighted)

Figure 12 shows the maximum pixel value of 7109 in the frame being pointed at the candle flame. The infrared camera took the frame from 64 feet away. This was the maximum range for fire detection observed when a candle was used as the fire source.

Stove test

Figure 13: Experimental setup using a stove

Figure 13 shows a similar kind of experimental setup for range testing using a stove as a fire source, placed 5 feet away from the camera. A trolley was used as seen in the picture to vary the distance.

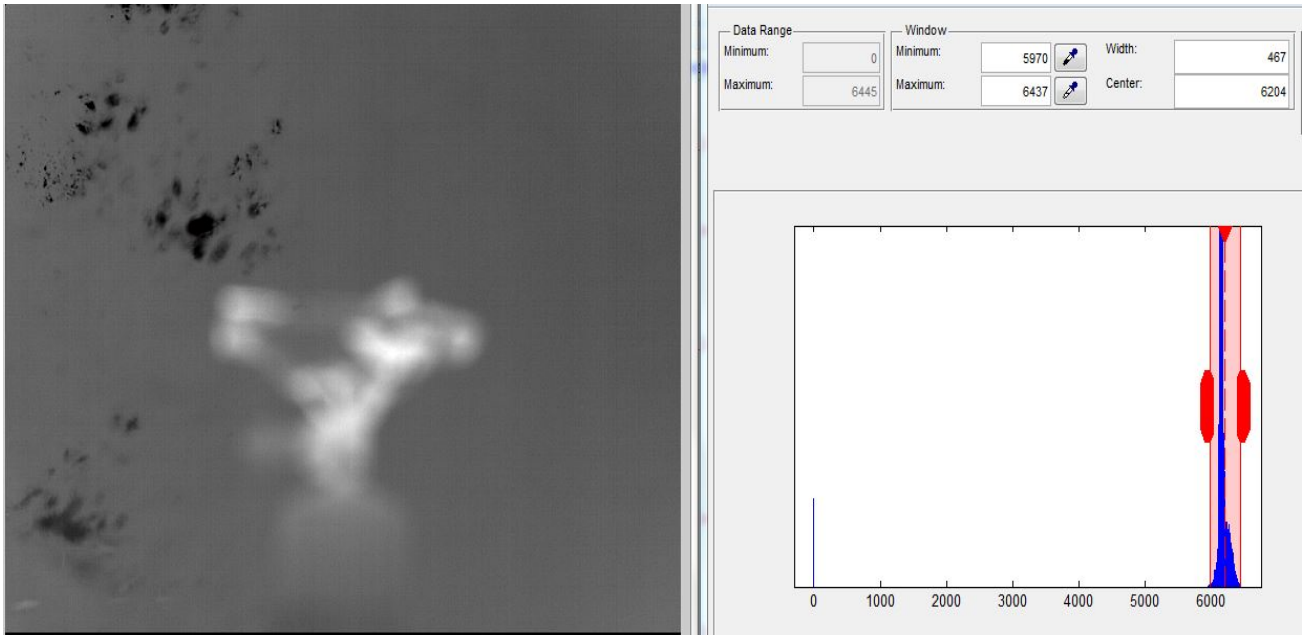
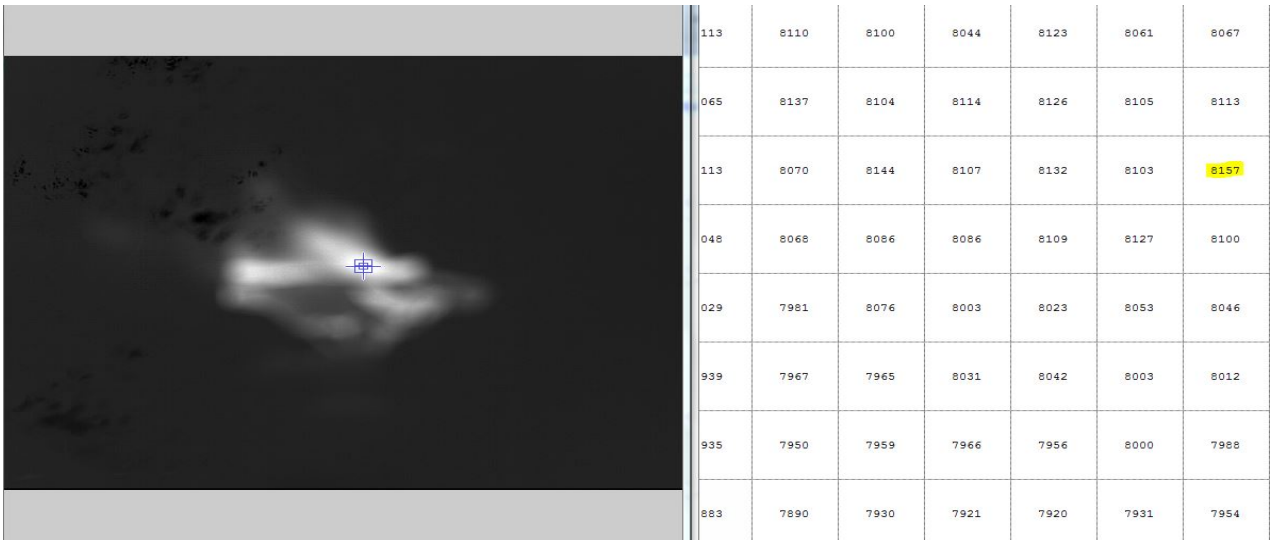


Figure 14: Infrared stove scene (5 ft. away) before fire showing the maximum pixel value

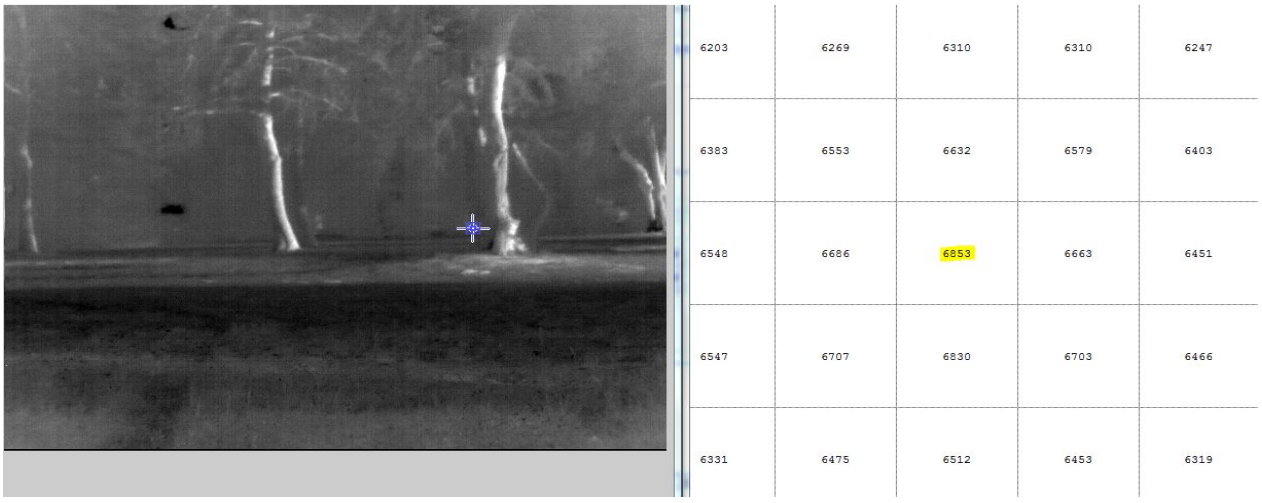
Figure 14 shows the infrared image of the scene with the stove, 5 feet away *before* being turned on. On right is the histogram of the image, showing the pixel range of the data in the frame. The maximum pixel value of 6445 can be read under the data range tab. Since the maximum pixel value in the frame is below the threshold (6800), no fire is detected. This is also obvious from the fact that there is *no fire in the scene*.



113	8110	8100	8044	8123	8061	8067
065	8137	8104	8114	8126	8105	8113
113	8070	8144	8107	8132	8103	8157
048	8068	8086	8086	8109	8127	8100
029	7981	8076	8003	8023	8053	8046
939	7967	7965	8031	8042	8003	8012
935	7950	7959	7966	7956	8000	7988
883	7890	7930	7921	7920	7931	7954

Figure 15: Infrared image of the stove (5 ft. away) with fire and pixel value chart (Max highlighted)

Figure 15 is the infrared image with the stove on, with the pixel value chart on the side corresponding to the pointer area in the image, aimed at the stove fire. Highlighted value in the table is the maximum pixel value of the frame, which is 8157. It's the same scene as shown in Figure 9 but the *fire is turned on this time*. Since the maximum pixel value of the fire is well above the threshold, fire is detected successfully and a warning email is sent right away. It is noted that the image is not focused but still the fire was detected as oppose to the candle experiment. This is due to the bigger size of the fire.



6203	6269	6310	6310	6247
6383	6553	6632	6579	6403
6548	6686	6853	6663	6451
6547	6707	6830	6703	6466
6331	6475	6512	6453	6319

Figure 16: Infrared image of the stove (232 ft. away) and pixel value chart (Max highlighted)

Figure 16 shows the infrared image of the stove with fire being 232 feet away from the camera with the maximum pixel value of 6853 as shown in the chart.

The trolley carrying the system was moved away from the stove until no fire was detected to find out the maximum range at which the system could detect relatively small fires. In this case the fire was detected, 232 feet away within a second and an automatic warning email was sent right away.

Newspaper Fire Test

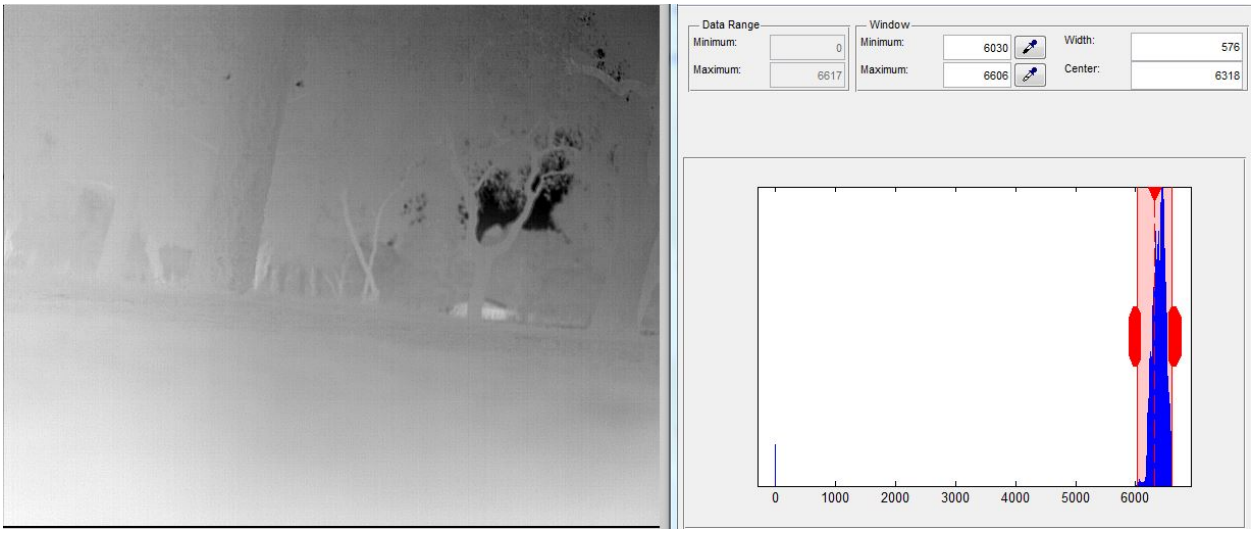
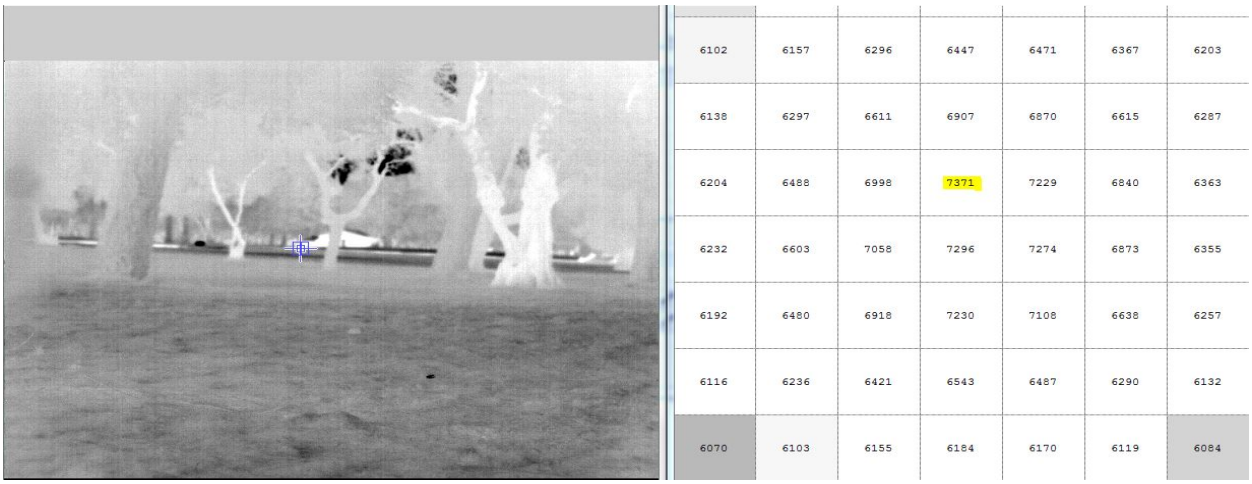


Figure 17: Infrared image of the scene for the newspaper fire with the pixel histogram

Figure 17 shows the infrared image of just the scene for the newspaper fire experiment before the fire was setup, with the histogram showing the range of pixel values in the frame. From the data range tab, we could see the maximum pixel value in the frame is 6617, hence in case of no fire nothing is detected.



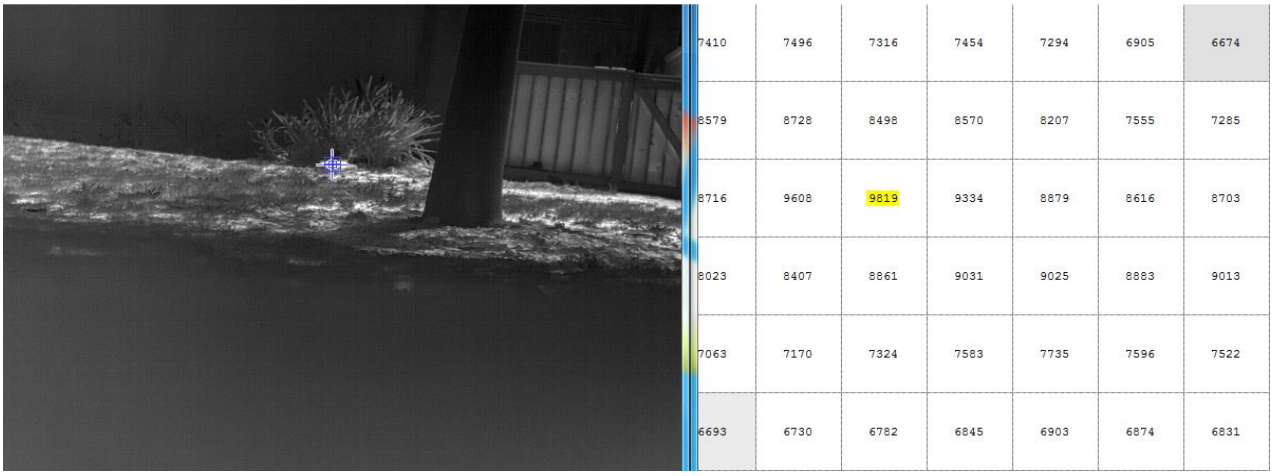
6102	6157	6296	6447	6471	6367	6203
6138	6297	6611	6907	6870	6615	6287
6204	6488	6998	7371	7229	6840	6363
6232	6603	7058	7296	7274	6873	6355
6192	6480	6918	7230	7108	6638	6257
6116	6236	6421	6543	6487	6290	6132
6070	6103	6155	6184	6170	6119	6084

Figure 18: Infrared image of the big fire (600 feet away) with the pixel value chart (Max highlighted)

Figure 18 shows the infrared image of the fire made out of burning newspaper, about double the size of the stove fire. The maximum pixel value of the frame is highlighted as 7371. Fire was detected roughly 600 feet away. When the fire was moved further no fire was detected.

Daytime vs. Nighttime experiment

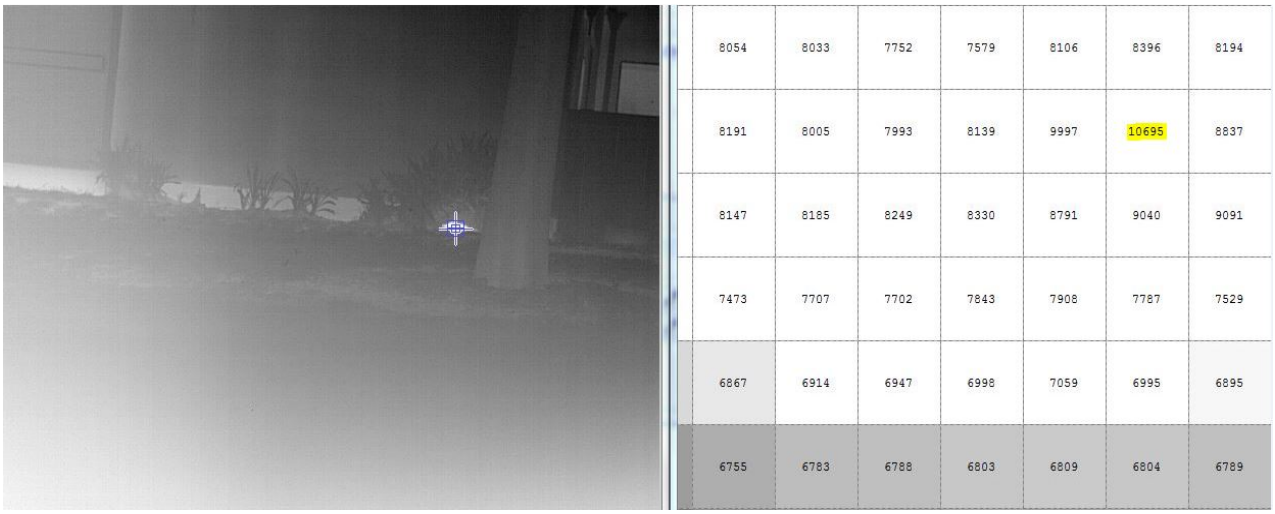
Another experiment was conducted to find out if the threshold of 6800 works well both during the day and night, as the surrounding temperature slowly changes. To explore this, an exact same setup was used to detect fire using a stove during daytime and then at night. The distance from the fire was kept exactly the same.



7410	7496	7316	7454	7294	6905	6674
8579	8728	8498	8570	8207	7555	7285
8716	9608	9819	9334	8879	8616	8703
8023	8407	8861	9031	9025	8883	9013
7063	7170	7324	7583	7735	7596	7522
6693	6730	6782	6845	6903	6874	6831

Figure 19: Infrared image of stove fire during Daytime with the pixel value chart (Max highlighted)

Figure 19 shows the infrared image of the fire with the pixel value chart on the side corresponding to the pointer area in the image, aimed at the stove fire. This image corresponds to the experiment during Daytime. The pixel chart shows the maximum fire pixel value of 9819 which is well above the threshold, thus fire was detected right away.



8054	8033	7752	7579	8106	8396	8194
8191	8005	7993	8139	9997	10695	8837
8147	8185	8249	8330	8791	9040	9091
7473	7707	7702	7843	7908	7787	7529
6867	6914	6947	6998	7059	6995	6895
6755	6783	6788	6803	6809	6804	6789

Figure 20: Infrared image of stove fire during Nighttime with the pixel value chart (Max highlighted)

Figure 20 shows a similar infrared image of the fire as earlier but during the night time with the maximum fire pixel value of 10695. The fire was detected this time as expected but an important thing to observe is that the maximum fire pixel value is a lot higher than the one observed during the daytime. Also, comparing both the images we can see that the nighttime image is more brighten up than the daytime. To further explore this, for instance the ground pixel values from both the images were compared by placing the pointer at some point on the ground other than the fire. The figures showing the comparison are as follow:

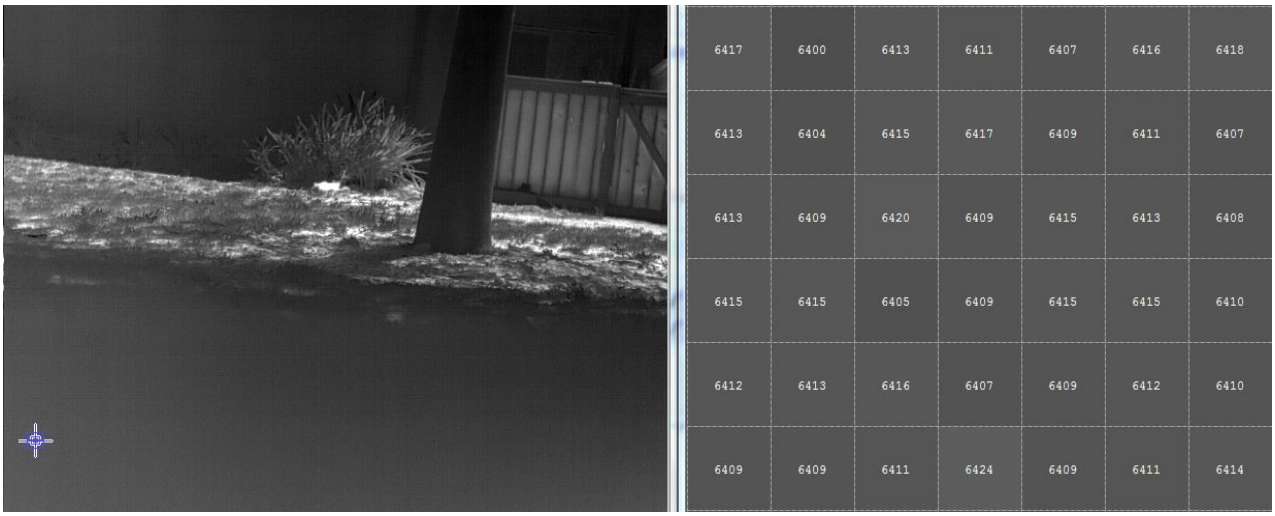


Figure 21: Infrared image of stove fire during Daytime with the pixel value chart (Max highlighted)

Figure 21 shows the same infrared image of the stove fire during daytime as Figure 19, with the pixel value chart corresponding to the pointer pointing at the ground. It can be seen from the chart that all the ground pixel values are close to 6400.

Figure 22: Infrared image of stove fire during Nighttime with the pixel value chart (Max highlighted)

Figure 22 shows the same infrared image of the stove fire during Nighttime as Figure 20, with the pixel value chart corresponding to the pointer pointing at the same ground location as in

Figure 21. It can be seen from the chart that all the ground pixel values are close to 6850 this time.

From the day vs. night experiment it can be concluded that the pixel values increase with the temperature going down, as the day progresses. The fire pixel values increased by roughly a 1000, and the ground pixel values increased by roughly 450 during the night. It can be explained through understanding the inner working of the long wave infrared camera. The camera basically contains an array of resistors and the value of each resistor corresponds to the individual pixel values in the frame. As the temperature goes down the resistivity goes up hence the camera response is higher at night when the temperatures are low. One might think that increase in pixel values is not uniform for the whole frame which is due to the difference in the heat level and heat properties of different objects in the frame.

Testing using a Telescope

In the last phase of testing a telescope was used as a lens for the long-wave infrared camera to see if we could increase the detection range farther than 232 feet. The experimental setup is similar to the one used for the stove fire earlier, at about the same time of the day but at a different location.

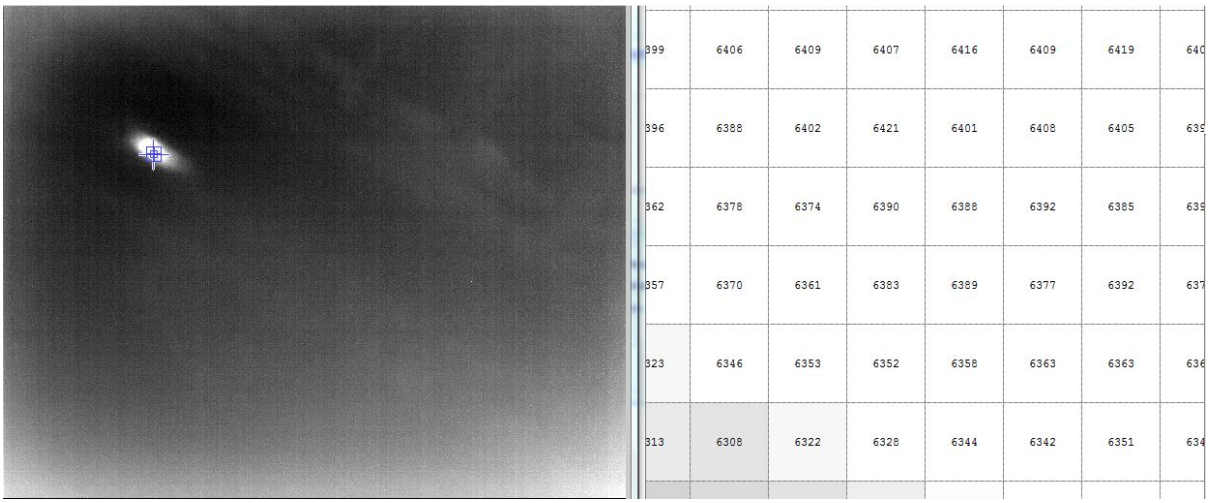
Prior to attaching the telescope we conducted the same stove experiment as earlier *for the second time*, keeping the system 232 feet away from the stove fire. The fire was detected at this distance and beyond this point no fire was detected. This confirmed our previous results.

This time a visual image of the scene with the stove fire was also taken as shown in figure 23 on the next page. As we can see no fire is observed in the scene with the naked eye.

Figure 23: Visual image of the scene with stove fire 230 ft. away

Next to perform the actual telescope experiment; a similar long wave infrared camera was mounted on the telescope using an interposer/connector borrowed from UCSB students.

The data and analysis using a telescope is as follows:



399	6406	6409	6407	6416	6409	6419	640
396	6388	6402	6421	6401	6408	6405	639
362	6378	6374	6390	6388	6392	6385	639
357	6370	6361	6383	6389	6377	6392	637
323	6346	6353	6352	6358	6363	6363	636
313	6308	6322	6328	6344	6342	6351	634

Figure 24: Infrared image of stove fire through a telescope (800 ft. away) and pixel value chart

Figure 24 shows an infrared image of the stove fire roughly 800 Feet away. It is observed that the fire pixel values are around 6400, which is below the threshold of 6800, thus no actual fire was detected.

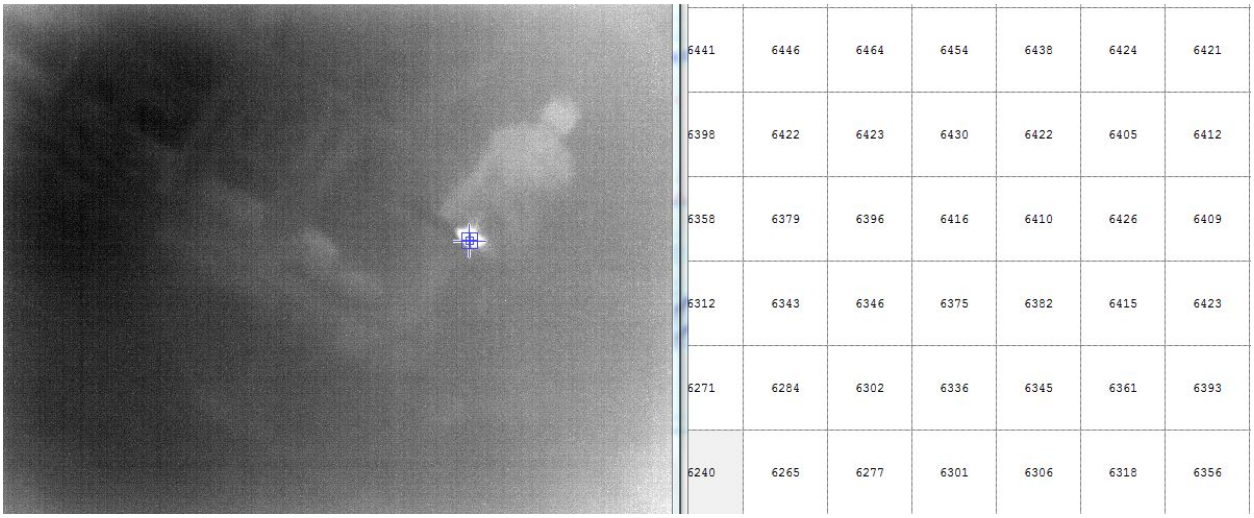


Figure 25: Infrared image of stove fire through a telescope (1000 ft. away) and pixel value chart

Figure 25 shows an infrared image of the stove fire moved up to 1000 Feet away. It is observed that the fire pixels are consistently around 6400 and again no actual fire was detected.

Although we know that no actual fire should be detected since the pixel values are below 6800, in both the above images the system *still detected fire as a false positive*. This is known from the fact that the maximum pixel value in the whole frame came out to be 6906, observed from the image histogram using the imtool in Matlab.

To further investigate, we moved the pointer in the frame until the location for the highest pixel value of 6906 was determined and is shown in Figure 26 below. It can be seen that there is only one pixel with that high of a value.

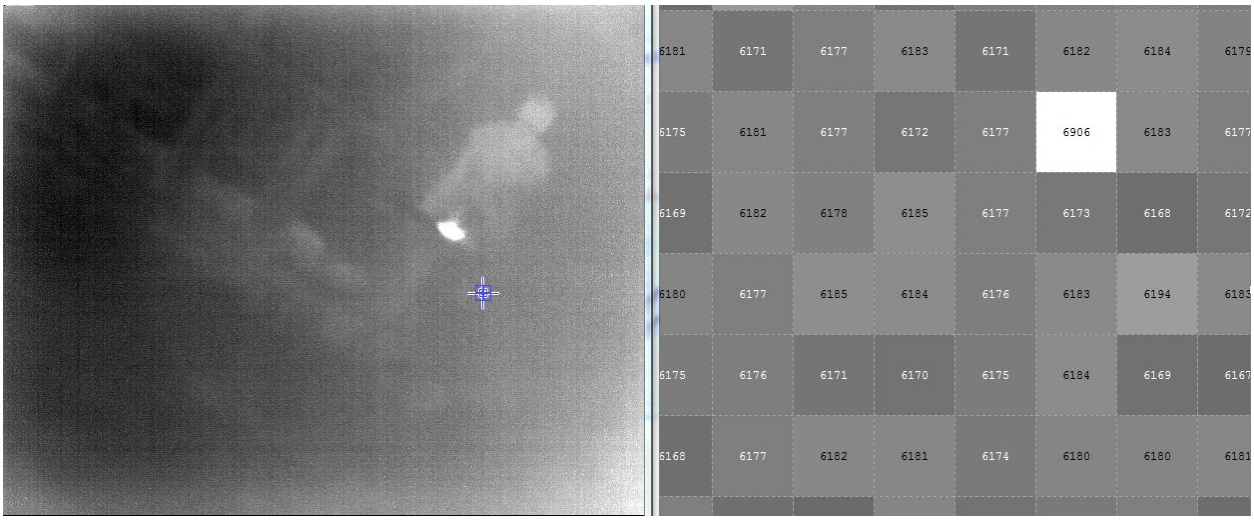


Figure 26: Infrared image of stove fire through a telescope with pixel value in the frame highlighted

It was realized that this pixel value is due to the fault in the new camera used with the telescope. As explained earlier the infrared camera is basically made out of an array of resistors whose value corresponds to the pixel values we see in Matlab. Each resistor is embedded in a substrate that determines the camera temperature and connects to a pixel reading IC board on the other side of the substrate. Sometimes if one of the resistors is constantly touching the substrate, its pixel value is going to be directly affected by the camera temperature and it's going to read a false value each time. This could be resolved either by replacing the camera or pointing the camera towards a black sheet and then figuring out which pixels are still reading values as oppose to being zero. Once those pixel values are determined, one can ignore those pixel values each time through software.

The reason for the fire pixel values to be well below the threshold when seen through the telescope is because the telescope consists of a mirror sitting in an Aluminum casing made out of Silicon dioxide which doesn't reflect infra-red energy well. Also, the Aluminum reflects up to 82% of infrared energy and absorbs the rest. The light reflecting from the mirror goes through a

prism before hitting the camera. Thus due to absorption of infrared light, less energy is being delivered to the camera, hence the overall pixel values are lower compared to just using a camera without a telescope. This could be resolved by lowering the threshold value when using a telescope or gold platting the mirror to reflect all the infra-red light.

Future improvements:

The future generation forest fire detection prototypes have a lot of potential and room for advancements. Some of the improvements for the Hardware and the Software are discussed below.

Software: In the past, there has been at least four Master's thesis, each exploring different fire detection algorithm techniques. Since this project was focused on building a real prototype, a simple threshold technique was used to detect fire. Thus next generation systems could integrate some of the advanced fire detection algorithms with the existing system by simply replacing the .m file containing the algorithm with a newer one. Rest of the code remains the same.

To overcome the anomaly of detecting fire at night with the same threshold as the day we could use an advanced technique of implementing a kernel around each pixel. The idea is to determine a background level at each pixel, as a reference. Then subtracting the background frame from the original frame will produce a "grassy" image, and fire will definitely be tall grass. More information could be found online as this technique is widely used in astronomical experiments.

Hardware: The experimental results point towards using a cooled camera weather it is mid wave infrared or long wave infrared as opposed to an uncooled one whose response changes with temperature as the day progresses. Since the cooled camera is more expensive and makes the system a lot heavy, it is a tradeoff between accuracy, portability and price.

The telescope used in this project to perform the last test, can highly increase the detection range using advanced algorithms or by lowering the threshold value. Also, gold platting the telescope mirror is another option. It could also be used to confirm the signs of fire by zooming in on the area where the fire is first detected by the infrared camera. In that case the telescope would only become operational when fire is detected by the infrared camera. A visual camera could be attached to the telescope to take a close up colored image of the fire/smoke. The telescope can also be programmed to turn around monitoring a wider area for fire signs.

Another improvement could be actually using a programmed IC board instead of using a big laptop to make the system even smaller.

Conclusion:

The goal of the project was to build a prototype system which allows us to look at a scene like a hill side and be able to detect fire. The system consists of an uncooled long-wave infrared camera, a network adapter, a small battery, a DC-AC converter, and a laptop. Also the camera can be attached to a telescope to increase the range of detection by lowering the threshold since the camera receives less infrared energy through a telescope; for instance a stove fire which can be detected up to 230 feet with just the infrared camera can be detected up to 1000 feet away with the telescope. In our testing the early forest fire-detection system has shown very promising results from detecting a small candle fire from 64 feet away to detecting a 1.5 feet newspaper fire from 600 feet away. As noted before, the range can be greatly increased using a telescope with slight variations in the algorithm. Figure 6 shows the overall flow of the software and how different pieces fit together. There was some discrepancy between the day and night time data; because this system uses uncooled infrared camera whose response varies with the ambient temperature. As noted in future improvements cooled camera or correcting this deviation in software using a technique like building a kernel around each pixel can get rid of this variation in day and night data. Evident from the testing results the early forest-fire detection system is a very promising prototype to prevent forest fires with a lot of room for future advancements.

BIBLIOGRAPHY:

Vickers, Virgil E. *Plateau Equalization Algorithm for Real-time Display of High-quality Infrared Imagery*, *Opt. Eng*. 35(7), 1921-1926 (Jul 01, 1996).<http://dx.doi.org/10.1117/1.601006>.

Inc., Flir. *Photon 640 Slow Video User's Manual* (n.d.): n. pag. Dec. 2008. Web. 2 Mar. 2014. <http://www.flir.com/uploadedFiles/CBI/Documents/Photon640_UsersGuide.pdf>.

"Legacy Camera Software." *FLIR Cores & Components Legacy Software*. FLIR Systems, Inc., n.d. Web. 2 May 2014. <http://www.flir.com/cvs/cores/view/?id=53130>.

APPENDIX A — ANALYSIS OF SENIOR PROJECT DESIGN

1. Summary of Functional Requirements

The goal is to implement a fixed land-based early forest fire detection system capable of detecting forest fire within a few seconds with range of at least 200 feet. Warning would be issued wirelessly upon detection of a fire. The whole system needs to be powered from a 12 Volt battery for automatic monitoring all day long. The whole unit needs to be enclosed in a casing to prevent damage to the camera and electronics in case of hazardous environmental conditions. Lon wave Infrared camera would be used as preliminary steps to help us detect the fire in different environments and at different times of a day. The warning would be issued based on fire detection to alert the fire department beforehand.

2. Primary Constraints

Significant challenges included implementing automatic frame grabbing software and complex Matlab algorithm to accurately detect fire. The range of detection depends on the fire size, and once the warning is sent the time to receive the email depends on the internet connection of the user, which sometimes might add up to a minute or even more, which is enough for the fire to grow. Another big constraint was the short amount of time to complete the project, since there is so much potential to make it better.

3. Economic

This project can potentially reduce fire damage in terms of money and property. It could help save government millions of dollars because fire in early stage can be easily contained and the property damage would be lot smaller. Also the fire causes lot of damage to the natural habitat, like trees, animals living in the area. All this can be reduced with an early fire detection system. The initial cost of the project is the infrared cameras, which cost up to couple thousand dollars. On top of that the cost of the battery and the cost of casing to protect the system would also add up. Benefits come only after the project is assembled and ready to use; then it can detect early signs of fire and reduce the monetary damage. Most of the parts used in the project have been donated by FLIR and the only thing we bought was the DC to AC convertor from Amazon for $26. But the parts are very expensive, just the infrared camera by itself cost around $6000.

4. Environmental

Preventing forest fire through small fire detection is the biggest environmental impact associated with the use of this project. This project uses forest ecosystems directly while sweeping the forest area with a thermal camera looking for signs of early fire. We specifically aim to preserve all the forest ecosystems which provide natural resources such as wood, many plant products such as fruits, nuts, latex for rubber etc. Forests also help in cleaning air and preventing floods which indeed adds to the welfare of human beings. This project has a very big impact on animal and plant life as it

intends to prevent forest fire by alerting the fire department beforehand whenever a small fire or a heat source is detected. Forest fires are an immense burden to the environment, animal and plant life. In 2012, the United States spent an excess of $1.2 billion dollars trying to suppress fires that claimed over 9.3 million acres of land and proved to be a catastrophe for all the animal life living there. Early detection of forest fire can drastically reduce this loss.

5. **Manufacturability**

The system is manually put together as a prototype and if we happen to mass produce it, additional labor costs would add up. A detailed cost analysis including case studies would be done, to determine a cheapest way to get parts in bulk to do a successful business. Intuitively, the cost per unit would go down if we happen to mass produce it as we will buy the parts at a discount in bulk.

6. **Sustainability:**

The biggest issue with maintaining the complete system is its battery. The battery needs to be recharged and after couple of years it needs to be replaced. Therefore the system is not sustainable because it needs a new battery every couple of years and the old batteries must be disposed properly. An improvement would be to design a project that doesn't need to be recharged, i.e. solar powered. The project would still require a battery but then the battery doesn't need to be manually recharged. The challenge with solar charging would be to find technology that would provide enough power to charge a 12V battery, without taking lot of space.

7. **Ethical:**

The project is design to help save lives and reduce the damaged caused by the fire. The system detects early signs of fire; therefore there is no way to misuse of this product.

8. **Health and Safety:**

Since the camera will be installed on a tower, the biggest safety concern would be the camera falling and hitting someone on the ground when in the field working. Since the system is mostly automated, there aren't any health concerns in terms of using the product.

9. **Social and Political:**

The political and social issue associated with the project is that forest fire every year does millions and millions in damage to property, and destroys the habitat for many animals. Therefore if our system can detect even 20% of the fires before it gets out of hand, then we can save many lives and homes. The project directly impacts the firefighters, people living in dry forest areas, and the government; because it can alert the firefighters early enough, who can then alert the near residences, and also save government money if the fires can be controlled in its early stages. Indirectly this project impacts all of us because fires destroy our natural resources, and since we pay

the taxes, the money that government spends on the fires also goes from our pocket. This project will benefit everybody because it will help save lives, and save natural resources.

10. Development:

This project teaches about the infrared camera technology and how efficiently it can be used to detect objects at distances. For a real life implementation of this product, cooled infrared cameras would do a more accurate job with higher consistency as the camera response would not change much due to temperature. Using a more sophisticated algorithm along with the telescope and some kind of rotation mechanism for monitoring all around would turn this system into a real life product.

APPENDIX B — Code

C++ program: Program used to control the camera

The code is much easier to read if the cpp file is opened in editor like Notepad++ because there are lots of comments and Notepad++ shows the comments in different color. There are parts that are commented out but still kept in code. The commented out parts were changed with help from Dr. Gary Hughes to make the program work with our camera. We are leaving the commented out parts so if in the future improvements the camera is changed, the next person working on the code can easily see where we made changes to make it work.

```
#ifdef UNICODE
#undef UNICODE
#endif

#include "stdafx.h"
#include <iostream>

#include <windows.h>
#include <fstream>
#include <comutil.h>
#include <stdlib.h>

#include <ISC_Camera.h>

#pragma comment(lib, "comsuppw.lib")

using namespace _com_util;
using namespace std;
const char outfilename[] = "IRCAM.fits";

int main(int argc, char* argv[])
{
	FILE * pFile;
	isc_Error err;
	int id;
	short rows, cols;
	BSTR  VIF[ISC_MAX_INTERFACES];
	BSTR  devices[ISC_MAX_INTERFACES][ISC_MAX_DEVICES];
	short *buf;
	USHORT *buf2, *buf1;
	int num_if, num_devices[ISC_MAX_INTERFACES];
	int i, j;
```

```
int frameSize;

cout << "Indigo Video Interfaces: " << endl;

for (i = 0; GetVideoIF(&VIF[i], i) == eOK; i++) {
        cout << ConvertBSTRToString(VIF[i]) << endl;
        for (j = 0; GetIFDevice(VIF[i], &devices[i][j], j) == eOK; j++) {
                cout << "        " << ConvertBSTRToString(devices[i][j]) << endl;
        }
        num_devices[i] = j;
}
num_if = i;
cout << endl;

err = OpenVideo(VIF[0], devices[0][0], &id);
if (err == eOK) {
        cout << "OpenVideoIF " << ConvertBSTRToString(VIF[0]) << " " <<
        ConvertBSTRToString(devices[0][0]) << " returned, Error: " << err << " Device ID:
        " << id << endl;

        err = GetFrameSize(id, 0, &rows, &cols);
        cout << "GetFrameSize returned, Error: " << err << "  Rows: " << rows << "  Cols: "
        << cols << endl;

        err = SetFrameSize(id, 0, rows, cols);
        cout << "SetFrameSize returned, Error: " << err << endl;

        err = SetPixelDepth(id, 0, eISC_16BIT);
        cout << "SetPixelDepth returned, Error: " << err << endl;

        err = StartVideo(id, 0);
        cout << "StartVideo returned, Error: " << err << endl;

        buf = (short *)malloc(rows * cols * sizeof(*buf));
        buf1 = (USHORT *)malloc(rows * cols * sizeof(*buf));
        buf2 = (USHORT *)malloc(rows * cols * sizeof(*buf));
        err = GrabFrame(id, 0, buf);
        cout << "GrabFrame returned, Error: " << err << endl;



        ofstream out(outfilename);

        if (!out){
```

```
        cerr << "Failed to open output file " << outfilename << endl;
        exit(2);
}

out << "SIMPLE  =                T / File Generated by UCSB IRCAM               ";
out << "BITPIX  =               16 / Data is actually 14 bits                   ";
out << "NAXIS   =                2 /                                           ";
/*              out << "NAXIS1  =              324 /                             ";
out << "NAXIS2  =              256 /                                           ";
*/
out << "NAXIS1  =              644 /                                           ";
out << "NAXIS2  =              512 /                                           ";
out << "BZERO   =                0                                            ";
out << "END                                                                    ";
for (i = 0; i<29; i++) out << "                                                 ";
out.close();

/*for ( i=0; i<rows; i++ )
{
for ( j=0; j<cols; j++ )
buf2[i*(j+1)]=buf[(i*(j+1))];/*FITS image data is usually flipped, and the word
                order needs to be reversed*/
/*}*/

/*
/* 89244 = 324*256, which is not Photon-640
Change to 644*512 for Photon-640
for ( i=0; i<82944; i++ )
buf1[i]=buf[82943-i];
*/
frameSize = 644 * 512;
for (i = 0; i<frameSize; i++)
        buf1[i] = buf[(frameSize - 1) - i];

for (i = 0; j<rows; j++)
{
        for (i = 0; i<cols; i++)
        {
                buf2[i + (j*cols)] = buf1[(cols - i) + (j*cols)];
        }
}
/* for ( i=0; i<82944; i++ ) */
for (i = 0; i<frameSize; i++)
{
```

```
                buf2[i] = ((buf2[i] << 8) | (buf2[i] >> 8)); /*Hard coded bit rotation, as the
                        camera is in BigEndian, and fits is LittleEndian*/
                /* buf2[i] = buf2[i] << 2; */ /*Added by Prince to deal with signed
                        numbers*/
                /* Probably not doing anything
                if (buf2[i] < 0)
                {
                buf2[i] += 32768;
                }
                */
        }

        pFile = fopen(outfilename, "ab"); /*Opens the file as binary*/
        if (err == eOK)
        {
                /* fwrite(buf2,2,82944,pFile); */
                fwrite(buf2, 2, frameSize, pFile);
                /* This looks like it is putting some blank rows?? Take it out for now...
                for ( i=0; i<576; i++ ) fprintf(pFile," ");
                */
        }
        fclose(pFile);


        err = StopVideo(id, 0);
        cout << "StopVideo returned, Error: " << err << endl;

        err = CloseVideo(id);
        cout << "CloseVideoIF returned, Error: " << err << endl << endl;
}


cout << "Indigo Control Interfaces: " << endl;

for (i = 0; GetControlIF(&VIF[i], i) == eOK; i++) {
        cout << ConvertBSTRToString(VIF[i]) << endl;
        for (j = 0; GetIFDevice(VIF[i], &devices[i][j], j) == eOK; j++) {
                cout << "       " << ConvertBSTRToString(devices[i][j]) << endl;
        }
        num_devices[i] = j;
}
num_if = i;
cout << endl;
```

```
	cout << endl << endl << "Hit 'Enter' to exit: ";
	/*cin.ignore();*/
	return 0;
}
```

Matlab Code:

A new email account was created for this project. The emai address is 'firecalpoly@gmail.com' and the password is 'firedetection'.

```
clear
i = 0;
while(1)
	%Running the .exe file
	system ('firedetection.exe')

	% Reading FITS image
	imagep=fitsread('IRCAM.fits');
	 imagep(imagep < 0) = 0;		%changing negative values to zero for displaying purpose
	image = flipdim(imagep, 1);	 %Flipping the image 180 since the camera images are inverted
	fitswrite(image,'IRCAM.fits');	%Overwrites the original flipped image with the flipped image

	%Renaming the file
	i=i+1;
	fname = sprintf('IRCAM%d.fits', i);
	movefile('IRCAM.fits',fname);

	%For Displaying images
	%imtool('IRCAM_hand_fix2.fits')
	%imageagc=agc(image,512,644);
	%imshow(imageagc)

	% Threshold fire detection
	image(image<6800)=0;
	image(image>0)=255;

	%detection algorithm
	if max(image(:)) == 255
		disp('FIRE DETECTED');
		disp('MAX PIXEL VALUE IN FRAME');
		disp(max(imagep(:)));
		imagesc(image)
		colormap(gray)

		%Automatic Email
		myaddress = 'firecalpoly@gmail.com';	% this email address was created for testing so
							anyone can use it
		mypassword = 'firedetection';		%password for the email above
```

```
                setpref('Internet','E_mail',myaddress);
                setpref('Internet','SMTP_Server','smtp.gmail.com');
                setpref('Internet','SMTP_Username',myaddress);
                setpref('Internet','SMTP_Password',mypassword);
                props = java.lang.System.getProperties;
                props.setProperty('mail.smtp.auth','true');
                props.setProperty('mail.smtp.socketFactory.class','javax.net.ssl.SSLSocketFactory');
                props.setProperty('mail.smtp.socketFactory.port','465');
                sendmail(myaddress, 'Gmail Test', 'Hello Fire Detection alarm');

        else
                disp('No Firedetected');
                disp('MAX PIXEL VALUE IN FRAME');
                disp(max(imagep(:)));
        end
end
```

APPENDIX C — Setup & User Manual

Software:

Following softwares must be installed on a laptop for the fire-detection system to be able to communicate with the laptop:

- **Photon SDK:** This will install all the drivers and the libraries needed to communicate with fire-detection system.
- **Coyote:** This checks to see if camera is connected to the laptop. Run this software before running Photon GUI to make sure that the camera is properly connected to the laptop.
- **Photon GUI:** This will allows us to look at the live output of the camera as long as the camera is connected properly.
- **Matlab:** This is required to take, store, and process data. Basically software side of the system depends on Matlab.
- **Notepad++:** This is an editor which allows for easy editing of the code. This is not required; one can choose to use their favorite editor.
- **Microsoft Visual Studio:** This is only required if modification of C++ code, which connects to the camera, is needed. Microsoft Visual Studio allows for easy changes to the code and compiling for an executable which is used with Matlab to take data from the camera.

Setting up the Laptop:

Before the camera can connect to the laptop, the IP address of the laptop must be set to static for the Ethernet adapter.

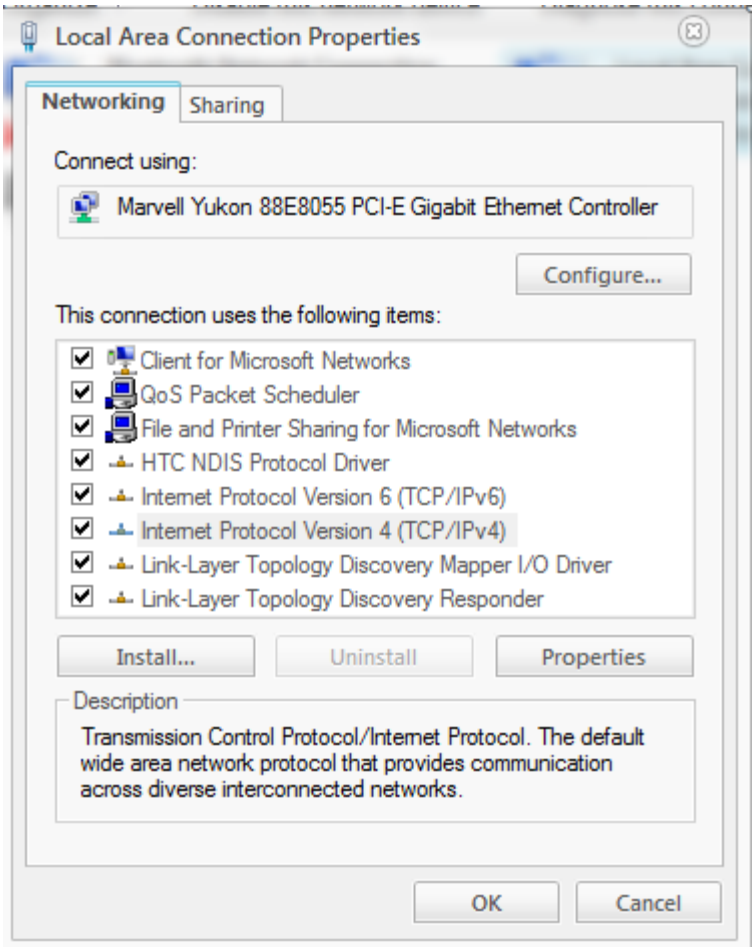


Figure 27: Properties of the Ethernet Adapter

To set the ip address to static, right click on the Ethernet Adapter under Network Connections and select properties. Figure 27 shows the properties dialog box for the Ethernet adapter. Then click on Internet Protocol Version 4 to make sure it's highlighted and click on properties. Figure 28 shows the next dialog box and set the values for IP address & the Subnet mask to connect to the camera as shown. This will allow the laptop to find the camera over the Ethernet.

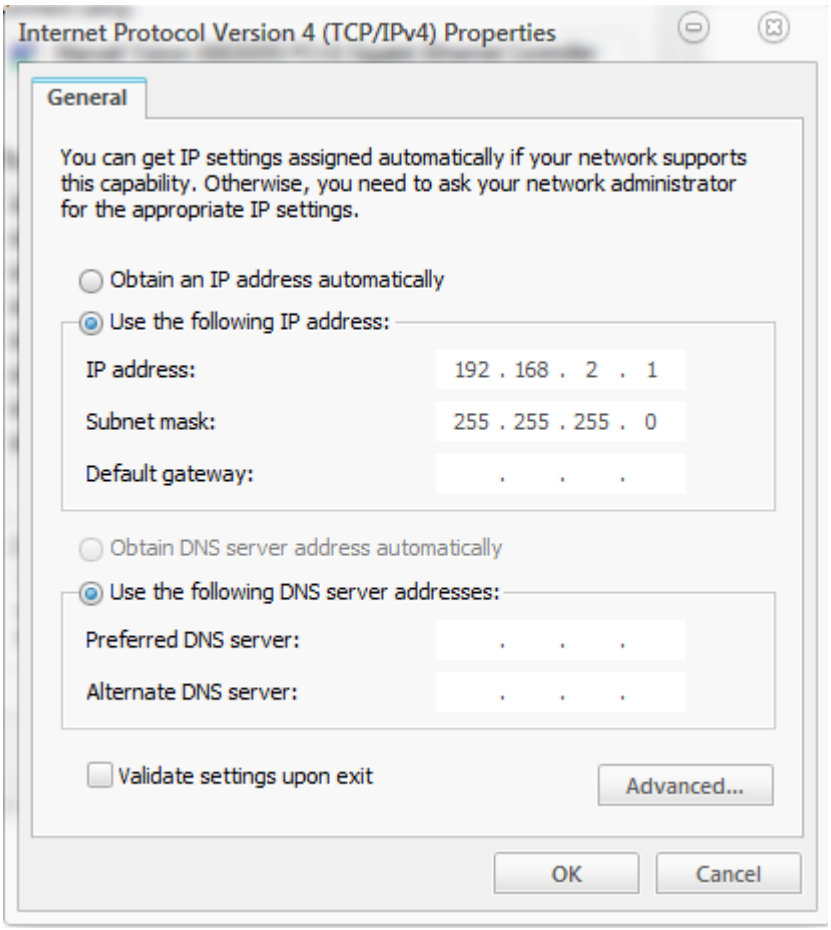


Figure 28: IPv4 Properties

Once the IP address is setup launch Coyote and hit "OK" on the first pop-up screen. A new window should pop up, and then click on "Detect". If the IP address configuration is setup correctly and all the wires are connected properly, then Coyote should detect the camera. Click "OK" and exit coyote. Coyote is used to check if the camera is connected; there is no need to add device here. Figure 29 below shows a sample of the Coyote application window.

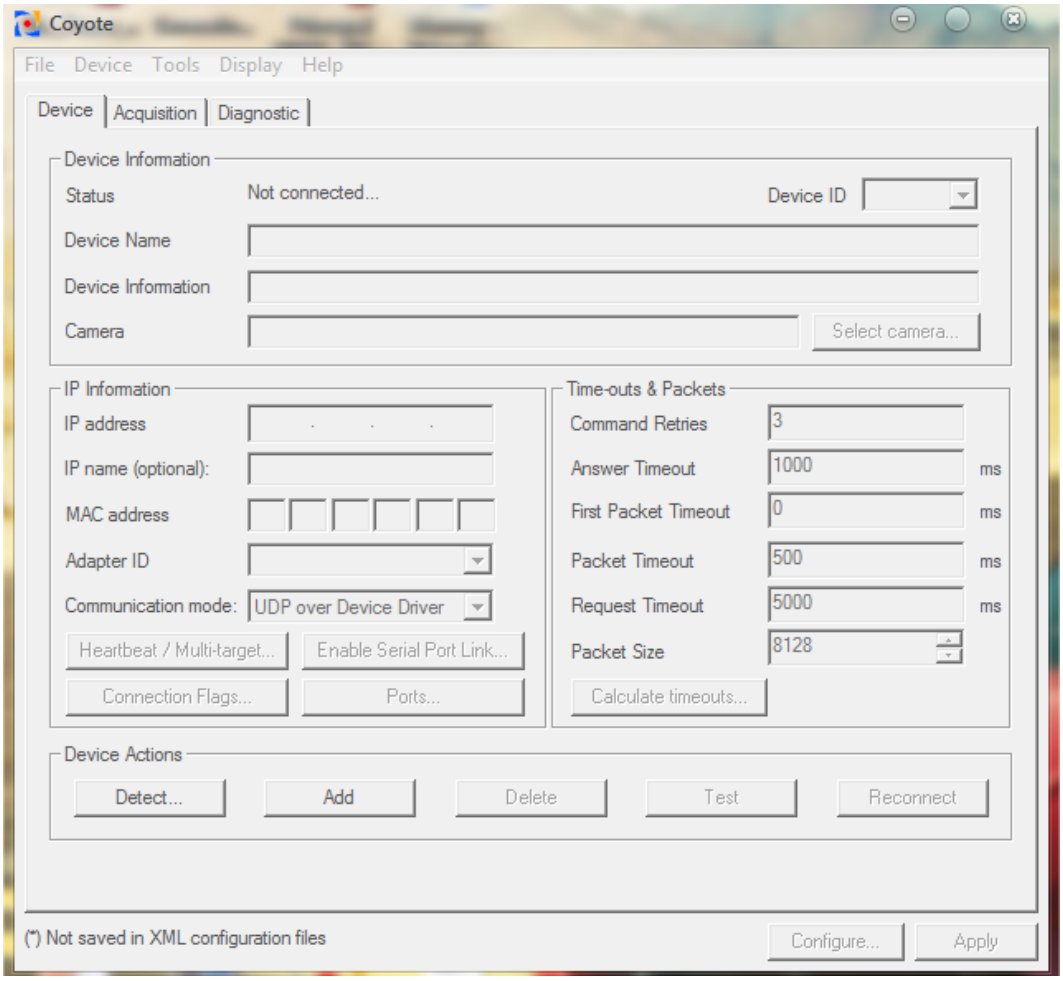


Figure 29: Coyote Application Window

Once coyote detects the camera, open Photon GUI to see the live video feed from the infrared-camera. Photon GUI can also be used to grab data but the data from Photon GUI is stored in TIFF file format. Photon GUI is a great tool for looking at scenes and pointing the camera in the right direction for testing. This concludes the setup of the software and connecting to the camera. After this one should be able to take data autonomously using the executable.