Video Processing and Feedback Team

Mohammad Sadeghian
Siva Teja
Mahmoud Tohmaz

1. Introduction

Drones have been an integral part of allowing us to see in places we normally can’t go. Manned aircrafts are limited in size and maneuverability. Additionally, due to the fact that a human pilot mans them, they cannot enter hazardous areas to observe and reconnoiter. However, drones solve these setbacks. Since they are no longer manned, they are much smaller in size, easily maneuverable, and can enter hazardous areas without putting harm to any human pilot.

One of the environments in which drones have a huge potential is in firefighting. However, the drone’s task is not in fighting the fire, but aiding in rescuing victims as well as map out the area for potential hazards for firefighters. Firefighters will later enter the building or area to attempt to rescue any victims or fight the fire. Rather than put the firefighter in harms way by letting them search room by room for any victims. The drone will enter first and scan the building for any potential victims while mapping out the safest route to that victim. This effectively minimizes the risk of injury for the firefighter and also increases efficiency since the drone can easily maneuver the building faster than a firefighter.

The main task of the drone is find victims as well as map out hotspots. To do this, we used an infrared camera to search for possible victims through the smoke and fire. We also used the infrared camera to search for hotspots, which would help the firefighter avoid these areas due to increased risk. In order to transmit the video back to the operator, we created a Wi-Fi connection between the drone and the operator. This allows us to stream the video back to the operator. The Wi-FI connection allows for great bandwidth to ensure the data reaches the operator with little latency.

2. Video Feedback and Control

A. Goal

There are many environments that humans and human controlled aircraft cannot safely enter. For these situations, a drone can be used to enter these locations without putting a human pilot at risk. The goal of this project is to implement a drone to aid firefighters in rescuing victims in burning buildings and to also map out hotspots which are a threat to firefighters due to weakened integrity. Our proposed design is to provide a lightweight video capturing system along with an onboard processing unit. This design will be capable of working under the stress of fire and smoke to detect hot zones and victims.

B. Video Feedback

I. Video Capture

Goal of video capture is to find a camera[11, 12, 13, 14, 15], which must meet the minimum requirement of ability to work under smoke and fire. It also must have enough resolution to track objects, be lightweight, have a USB interface[6] as it would be connected to onboard processing unit, and have a low cost. Cameras that work in visible light range will meet the specifications of resolution, weight, USB interface[6], and cost but they cannot work in smoky conditions. Their low wavelengths will inhibit them from moving across smoke particles. Infrared cameras with higher wavelengths (4-7μm) are only suitable for smoky environments. But IR cameras with these resolutions are quite expensive to buy with a starting price of $1000.

After an extensive search, we found IR cameras that are under $1000 but they needed a smartphone to capture image and video. Additionally, they have very low resolution because of their low cost. FLIR one and Seek Thermal[2] are the two options that are available for IR cameras. We preferred to use Seek Thermal[2] camera in our project as it has relatively higher resolution compared to FLIR one and also was less expensive. It only works with a smartphone and cannot be connected directly to a processing unit. But cost of the total unit and payload of the system will increase if smartphone is also included in the project. It would be a big challenge to read image frames directly from the camera, as it would reduce both the cost and payload of the system. Seek Thermal[2] camera is a proprietary device so there is absolutely no documentation available for these devices. The process of capturing images from this camera is discussed below.

Any USB device has a vendor ID and product ID associated with it. These details can be obtained when 'lsub' command is run in Linux terminal with the device connected to that machine. A host computer, to which a USB device is connected, uses these vendor ID and product ID details to initiate the correct drivers for that device. Once the drivers are running, application can proceed to communicate to the device directly. Seek Thermal[2] camera has a micro USB interface[6]. Since it is a proprietary device there are no device drivers available for Seek Thermal[2]. Vendor ID and product ID can be known with the help of 'lsusb' command stated above.

PyUSB is a package written in python with a useful set of function calls that will help to communicate with any USB device connected to host computer. The device that the PyUSB package is communicating is decided based upon the vendor ID and product

ID used in the python program. This package has function calls starting from finding the device with specific vendor ID details, configuring, setting up, sending control commands, reading, and writing data to device.

We used “Total Phase Data Center” protocol[8] analysis software to determine the USB protocol[6, 8] that Seek Thermal[2] camera is using. This software dumps out a binary file with all the device enumeration details. From these dumps we are able to figure out control commands and message strings to be used to access the setup and access the data frames from the device properly.

PyUSB package has a core module, which contains most of the functions we used to capture frames from camera. Seek Thermal[2] has vendor ID = 0x289d and product ID = 0x0010. With the help of these two IDs an instance of the device can be created when it is connected to computer/processing unit.

PyUSB package provides various function calls to communicate with the device. We mainly used two function calls “ctrl_transfer” which will tell the USB device whether the command is an initialization command or a command for read. The other function call used is the “read” which does a bulk data transfer in this case from USB device to application.

Next section describes the types of image data received from the IR camera and the processing algorithms used to reduce the noise and map the temperature regions.

II. Video Processing

For easy integration between video capture and video processing we used OpenCV python software for image processing algorithms. OpenCV python is just a wrapper around OpenCV C++ libraries, which enables calling image processing functions written in C++ directly from python. As it is just a wrapper around C++ libraries, there will not be much impact on memory usage when the python application is running.

Image data is read as strings when read command is issued using PyUSB. With python “Image” module this string is converted into python “Numpy” array. “Image” and “Numpy” python modules used to process images and do arithmetic operations respectively. Once image is stored as “Numpy” array it can be used for processing in OpenCV.

When read command is issued to Seek Thermal[2] camera, it returns 10 different kinds of frames. In the received frame a status field is set which determines the type of received frame. Out of the ten received frames we are able to determine the type of only three frames. Frame with status set to 1 is calibration data, status set to 3 is image data and status set to 10 is pre-calibration data. IR camera sends a calibration frame in between image data frames. Calibrated image is subtracted from image data to obtain the final image.

This final image obtained from the camera is found to be very noisy. The image contains a lot of dead pixels. The image below shows the image data that is seen after subtracting the calibration frame from it.

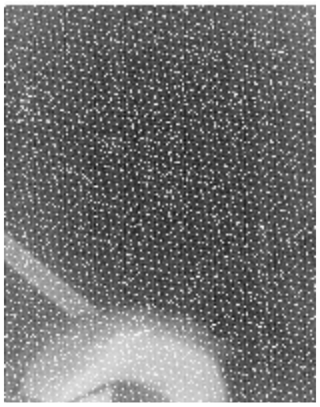

Figure 1: Image captured from IR camera

To remove the extra noise from the image, median filtering is applied to the image. This filter technique runs through every pixel and replaces the pixel value with median of neighborhood pixel values. In this way pixels with high intensity causing noise are replaced by pixels values from neighborhood. OpenCV has inbuilt median filtering function and 3x3 median filter window is used. Image obtained after applying median filtering is shown below**.**

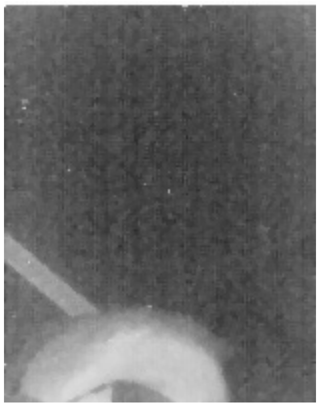

Figure 2: Noise filtered image using median filter

After filtering the image using median filter, a two-step threshold is done on the image. Threshold values are determined using relative intensities of pixel values. First threshold will reduce the gray scale image into binary image with white values corresponding to the regions with low temperature regions in the image. This binary image is passed to find contours function which will find all the closed contours in the image. Either rectangular bounding boxes or a line around the contour can be drawn using OpenCV functions. These bounded boxes are mapped from binary image to gray scale image. For lower temperatures the bounding box is green. The same procedure is followed for second threshold but the bounding boxes are drawn in red indicating higher temperatures.

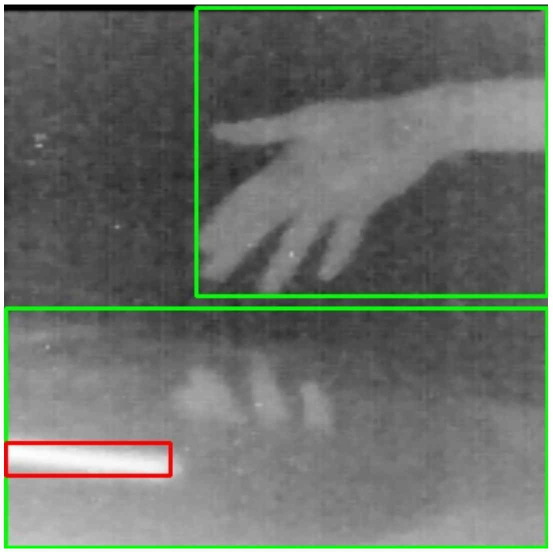

Figure 3: Output of video processing. Red box is laptop heat sink and green boxes denotes lower temperature

In the above picture, the red box is pointing to the region of laptop heat sinks. Green boxes represent regions of relatively lower temperatures. The images onto which both the red and green lines are drawn after processing will the streamed to the user as a feedback. Final images are saved to a standard www location and the webserver loads this image on its webpage which user sees at his end when connected through Wi-Fi.

B. Streaming

The goal of this portion is to provide a remote pilot with the ability to view the drone's cameras[1,5] with minimal latency. First, we had to create a connection between the drone and the pilot. We needed good bandwidth to stream the video, and good range to ensure a stable connection. To do this we decided to create a WiFi access point on the drone to allow users to connect to the drone. The reason behind choosing WiFi was due to the fact that streaming video from a camera[1, 5] required a lot of bandwidth. We looked into other modules, such as XBee, to transfer data between the drone and pilot. Although XBee boasted a far transmission range (boasting between 6-15 miles range depending on the module), the data transmission rate was very low to what we needed (a mere 200 Kbps). WiFi was the best option with the perfect tradeoff between range and data rate. This all depends on the equipment, for example the antenna which transmits the connection. But on average WiFi has a 30 meter range, which can be increased with better antennas and equipment. The data transfer rate is much better at 82 - 200 Mbps which is more than enough for streaming videos.

To create the WiFi access point, there has to be a small computer on board the drone. We used and tested both a Beaglebone black (BBB)[4] and Raspberry Pi (RPI) and therefore it is recommended that you use the same to ensure the same results. Creating the WiFi access point is relatively easy and the same between the two boards. The only issue with the BBB[4] is the compatibility with certain hardware. Not all WiFi USB adapters work properly on the BBB[4], so make sure they are compatible. You can check from the available list on the practical problems web page for a list of compatible BBB[4] Wi-Fi adapters[3]. Also, make sure the 5V adapter is plugged in to ensure the BBB[4] is given enough power to run the adapter[3].

Note: The BBB[4] only has one USB port and our USB hub wasn't supported by the BBB[4]. A list of supported USB hub's for the BBB[4] can be found on our practical problems web page. We weren't able to test the streaming with the Wi-Fi access point due to the fact we required multiple USB ports (3) to connect the cameras and Wi-Fi adapter[3]. Therefore we guarantee that the steps will work with a RPI, but not with a BBB[4] and may require a little more tweaking. We tested it together on the RPI and therefore can confirm it works coherently on the RPI.

In order for the WiFi access point (AP) to work, the proper packages must be installed. The packages required are hostapd and udhcpd. To install them, type the following code into your RPI or BBB[4] console (make sure ethernet is connected and working): sudo apt-get install hostapd udhcpd. Once these packages are installed, DHCP must be properly configured. We have already provided the proper files in our directory. Just copy them over as they are into their proper directories in the RPI or BBB[4]. Once everything is properly copied over, the AP must be started. Run the following commands separately to initiate both hostapd and DHCP respectively:

```
sudo service hostapd start
sudo service udhcpd start
```

The AP should have started up and can be seen on other devices as "drone." To automatically start the AP on boot type the following into the console (don't worry, it won't mess with an ethernet connection if you decide not to plug the wireless adapter[3]):

```
sudo update-rc.d hostapd enable
sudo update-rc.d udhcpd enable
```

Once the AP is up and running, the streaming can then be implemented. The idea behind this is that any user can connect to the WiFi access point “drone” (or any name that you prefer) and can then access an index file, which contains the stream. This way the user is not required to have a specified device and can be run on any platform as long it has a proper internet browser. The best package to stream a video is mjpg streamer. This allows you to stream a video at a good fps rate of above 20 fps. This is also the ideal package since it allows you to have onboard processing or to push the video from the drone to another more powerful computer to allow for improved video processing if there is a strain on the processing on the drone. First the relevant packages must be installed. Run the command below to install the libraries that are required for MJPG-Streamer:

```
sudo apt-get install libjpeg8-dev imagemagick libv4l-dev
```

Once the packages are installed, there is a missing videodev.h file that needs to be added. This file has been replaced by videodev2.h and running the command below will fix this issue:

```
sudo ln –s /usr/include/linux/videodev2.h
/usr/include/linux/videodev.h
```

Once the issue is resolved, MJPG-Streamer can now be downloaded using the command wget in linux. The download file link can be found in our software page. If this doesn’t work, most likely the download link no longer exists. To fix this issue you can search on Google for another download link and replace it in the wget command.

After the download is complete, unpack the source code using this command:

unzip mjpg-streamer-code-182.zip (The file name might be different)

To get MJPG-Streamer fully functioning, the proper plugins must be built using the following commands:

```
cd mjpg-streamer-code-182/mjpg-streamer
```

```
make mjpg_streamer input_file.so output_http.so
```

Now MJPG-Streamer can finally be installed. Use the following commands to complete this process:

```
sudo cp mjpg_streamer /usr/local/bin
sudo cp output_http.so input_file.so /usr/local/lib/
sudo cp -R www /usr/local/www
```

To start MJPG-Streamer use the command below. The –f option specifies the directory where the video is located. Make sure that is correct.

```
LD_LIBRARY_PATH=/usr/local/lib mjpg_streamer -i
"input_file.so -f /tmp/stream -n pic.jpg" -o
"output_http.so -w /usr/local/www"
```

The video will be saved to this location after processing. So make sure that the video feedback saves the video to that directory. You can now watch the stream by connecting to the "drone" Wi-Fi network and entering http://<IP-address>:8080 (IP address is 192.168.12.1 if you use our configuration files) to view the stream.

3. Problems Faced

A good number of issues are seen when capturing images from Seek Thermal[2] camera. When the temperature range captured in a given scene is very high, a proper mapping of temperature regions is not seen. This is illustrated in below image.

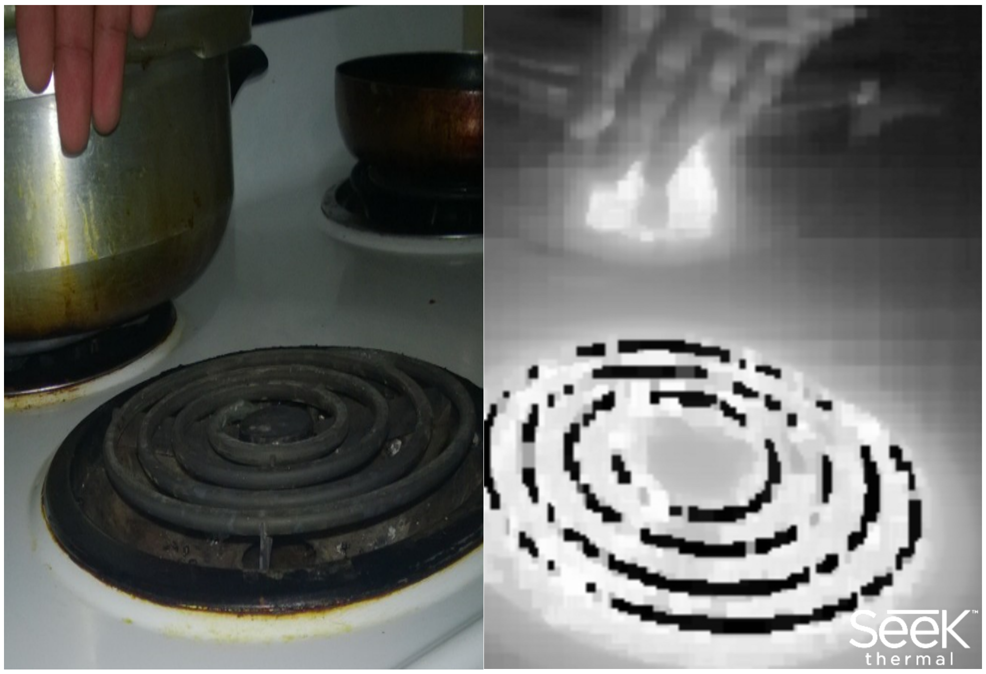

Figure 4: Left image captured using regular camera

Right image is same scene captured using Seek Thermal camera and smartphone

This image is captured directly from the smartphone. The picture is of coil of an electric stove, pressure cooker and human hand. As can be seen in the image, human hand is mapped to dark pixels indicating low temperature but the temperature of hand is more than that of the pressure cooker. The camera is also picking up the reflection of the hot object on the metal surface which in real situation is not hot. This can be seen in the image from Figure 3 also, where the reflections of hand are seen on the laptop. Also the hot coil is mapped to black at some portions of coil indicating low temperature where it is actually at very high temperature.

Also Seek Thermal[2] has a gradient issue. When the camera is pointed to a flat surface with uniform temperature all over the region, the camera still shows some gradient in the temperature mapping. A nice explanation of this issue can be seen in a link[9, 10] provided in the practical problems web page.

4. Operator Control

Unfortunately, we did not get any time to fully develop the operator control portion. However we have done some research on this matter. The easiest method is to use an RC module to control the drone, however one idea we had was to control the drone via mobile phone. We started developing a mobile app which allowed a user to control the drone. However, once we received the thermal camera we prioritized developing the video feedback over the operator control due to the fact that the RC module is sufficient (although not optimal) to controlling the drone.

5. Performance Results

A. Video Feedback

Seek Thermal[2] camera sends one calibration frame after a certain number of image data transfers to make sure that image data is being read properly. Reading an image data frame takes about 80ms. But reading calibration frame from camera takes about 500ms. Processing of each frame requires 40ms. So the maximum FPS that can be obtained from the IR camera is 8FPS. But when a calibration frame is sent from IR camera then FPS can be lower.

IR camera has inbuilt resolution of 206x156. This number of pixels in the image can be increased using image processing techniques in OpenCV without losing the details in the image. Python program written to process the video from IR camera resizes the image from 206x156 to 400x400 and the same image is written to output folder for streaming.

Since the IR camera has low resolution it will be difficult to map temperature regions that are beyond 5mts from the camera. Objects beyond 5mts are show in a very small region in the image and algorithm treats them as noise. This noise consideration for small regions will improve the algorithm performance for detecting objects which are at distance less than 5mts. This 5mts number is obtained by experimentation.

Python program when run on Raspberry Pi microcontroller consumes 60MB RAM at peak 10% CPU utilization to process the images from the IR camera.

B. Streaming

We tested how much RAM, FPS, and computer processing the streaming took. With the camera and streaming running, it only used less than 10 MB of RAM and can run efficiently on 100 Mhz of processing or more. The Wireless connection speed depends on how far the drone is from the display/user. We tested it inside the EI Garage, which is in a basement, and could transmit the signal within 10 meters with obstructions. A better wireless adapter will also cause an increase in range and connection speed. The FPS of the video goes above >20 fps which is more than enough to pilot the drone. However, this is constrained by the processor. If the processor is under stress you will see a drop in FPS.

6. Conclusions

With python code written using PyUSB package thermal images can be captured directly from the Seek Thermal[2] camera using an USB converter. A use of smart phone is not mandatory to capture the frames thereby cost and payload of the system is decreased.

OpenCV python code does not create much overhead for processing image frames and the FPS for processing depends on the time to grab the frames. Based on the performance of the Seek Thermal[2], a maximum of 8 FPS can be obtained for processing image frames.

Since streaming is done over a simple Wi-Fi connection any user with Wi-Fi enabled device, browser and display can easily connect to the Wi-Fi hotspot and stream the video.

7. Recommendations and Future Work

Much of the time was spent on getting to capture frames directly from the camera without the use of smartphone. Though capture from the camera was successful, image captured contains a lot of noise. Not many filtering techniques could be explored which will increase the detail in the image. There is lot of scope for improving the image details by noise reduction.

8. References

A. Components:

1. http://www.logitech.com/en-us/product/hd-pro-webcam-c920
2. http://obtain.thermal.com/category-s/1818.htm
3. http://www.amazon.com/Tenda-W311M-150Mbps-Wireless-Adapter/dp/B006GCYAOS
4. http://beagleboard.org/BLACK
5. http://www.microsoft.com/en-us/windows/compatibility/CompatCenter/ProductDetailsViewer?Name=Microsoft%20LifeCam%20HD-3000%20Webcam&vendor=Microsoft&Locale=1033,2057,3081,4105,16393&ModelOrVersion=T3H-00014&BreadCrumbPath=lifecam%203000&LastSearchTerm=lifecam+3000&T

ype=Hardware&tempOsid=Windows%208.1
6. http://www.beyondlogic.org/usbnutshell/usb1.shtml
7. http://www.eevblog.com/forum/testgear/yet-another-cheap-thermal-imager-incoming/
8. http://www.totalphase.com/support/articles/200424386
9. https://www.youtube.com/watch?v=u20NvzUX8wE
10. https://www.youtube.com/watch?v=8Cr8oZck5m8

B. Second Choice Components:

11. http://www.melexis.com/Infrared-Thermometer-Sensors/Infrared-Thermometer-Sensors/MLX90620-776.aspx
12. http://therm-app.com/product/therm-app-device/
13. http://www.flir.com/flirone/
14. http://www.adafruit.com/products/1030
15. http://www.adafruit.com/products/814

C. Algorithms and Methods:

16. http://en.wikipedia.org/wiki/Image_segmentation
17. http://cosmos.ucdavis.edu/archives/2013/cluster10/VISWANATHAN_SIDD_Paper.pdf
18. https://sites.google.com/site/practiceelectronics/projects/quadcopter
19. http://eprints.uthm.edu.my/2926/1/mongkhun_qetkeaw_1.pdf
20. http://pharos.ece.utexas.edu/wiki/images/1/1c/SystemOfQuadcopters-2012.pdf
21. http://eprints2.utem.edu.my/9339/3/Quadcopter_Rembedded_Controller_For_Altitude_Hold_And_Surveillance_-_24_Pages.pdf
22. https://sites.google.com/a/ncsu.edu/firefighting-drone-challenge/
23. https://www.youtube.com/watch?v=2i2bt-YSlYQ
24. http://www.lirtex.com/robotics/fast-object-tracking-robot-computer-vision/