Individual Lab Report #3

Akshit Gandhi

Team H (PhoeniX)

March 6, 2019

Team Mates:

Shubham Garg
Parv Parkhiya
Zhihao Zhu

Individual Progress

As promised in the last progress review, I began working on the thermal camera along with Steve. The task was to segment out regions with high temperature in the images obtained by the thermal camera. The tasks performed by me since the last progress review are outlined in more detail below.

FLIR BOSON 320 ROS Wrapper:

The basic step of working with any camera is to know how to get the data from the camera into the processing unit. The thermal camera by itself does not ship with any SDK or ROS wrapper that can help us to get the images. There are several Open-Source ROS packages available online, but they were not compatible with the camera model that we had. In our sponsor Sebastian Scherer's AirLab we had a student working on the same camera and he had a driver ROS package written for it; so we got the driver software from him and we started to get the images from the camera being published to the /thermal ROS topic.

Thermal Camera calibration:

Calibrating a normal RGB camera involves the use of a checkerboard pattern which gives you the intrinsic calibration parameters. But the thermal camera is different, not only it requires intrinsic calibration for the focal length, skew, distortion, etc parameters but it also needs to be calibrated based on the temperature. The process of calibrating the thermal gains of the camera can be found here. Basically the process I followed was:

1. Plug the camera into a Windows machine with Boson App installed.
2. Go to the calibration option and select start button
3. The GUI will prompt you to place a heated black body in front of the camera, so I placed a soldering iron in front of the camera such that the heated body occupies most of the image.
4. The camera will take some sample pictures and then the GUI will prompt you to place a black body which is 20 degree Celsius lower in temperature that the heated body. So with the soldering iron in the lab we have a precise control on the temperature so I was able to get exactly 20 degree difference in temperatures and let the camera take a few pictures.
5. After the camera sensor settled on the temperature, the camera calibration stops and now one can use it to detect temperature.
6. There are some additional calibration tools that need to be run in order to get a perfectly calibrated image so those are explained well in the document shared above.

This is how the calibration process was completed.

Region of Interest Segmentation:

Approach 1:

The first approach was to convert the grayscale images into a colorful image and segment the images based on the color but it turned out to be a bad idea as the thermal images already encoded that information with the intensities and using another layer of preprocessing was a waste even though it would make the images look good visually but it does not add any value to the solution. Some sample processed images are shown below.

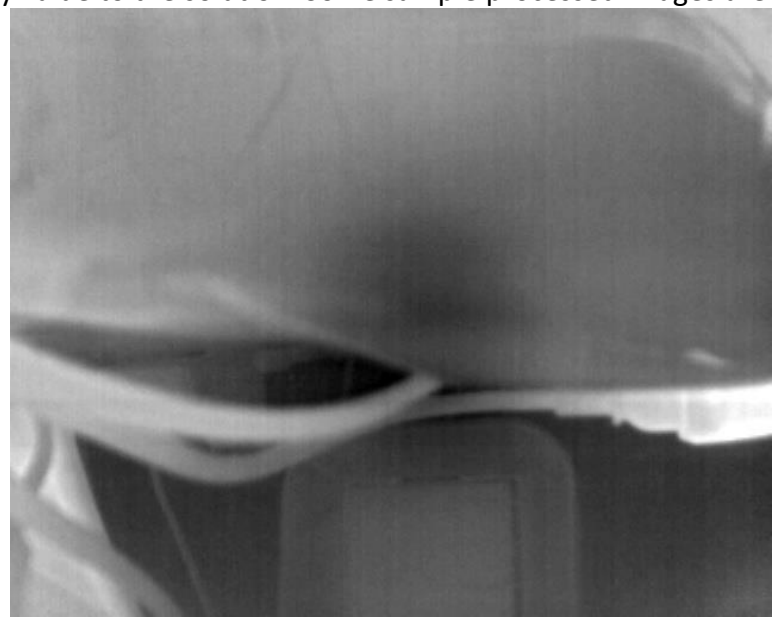

Figure 1: Input Image

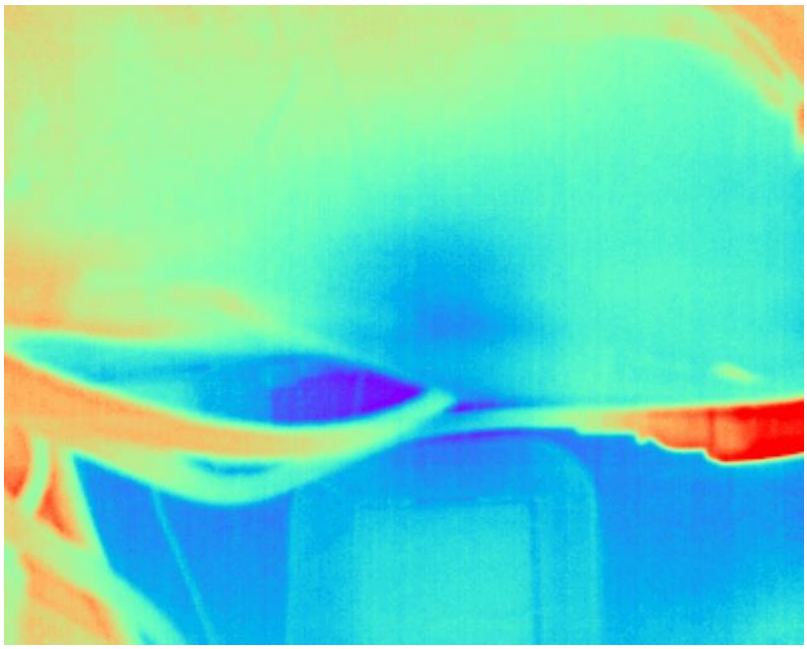

Figure 2: Grayscale image converted to color based on intensities

Approach 2:

The approach taken by me was to use the useful information given by the camera which was the intensity information which corresponds to the temperature of the object being seen by the camera. The intensity of 255 means that the object is the hottest in the frame and intensity of 0 means that it is the coldest object. So, for the competition we are not aware of the temperature of the simulated fire, but our sponsors told us to work on a regular shaped simulated fire which has significantly high temperature compared to the surroundings. Thus, I decided to segment the image based on a threshold. I read multiple strategies to find dynamic threshold and finally settled upon Otsu thresholding for my application.

The OTSU thresholding works to find separate the data into a histogram with two peaks such that the inter class variance is higher compared to the intraclass variance which leads to a binary image. The algorithm fails because it sets up a dynamic threshold for every frame and we don't want to detect objects in a scene that are not having any intensity of the simulated fire that we are looking for. So, I changed the approach and used a hard-coded threshold that only gives out regions which are of the temperature range that we are looking for.

The threshold rejects most of the outliers and now the strategy was to delete even high intensity regions that are at a high temperature but not the simulated fire. So, I got the area of all the connected components in the image and applied a threshold on the area so that smaller regions like lights and pipes carry hot air/hot water don't show up in the resultant image. After this suppression, I approximated a bounding box over the regions that are left after the previous processing and these are the regions where we think the fire is. We tried out bunch of strategies and the results can be seen in this video where the video in the 1st cell is the input image, 2nd cell is the output from the watershed algorithm, 3rd cell contains the algorithm developed by Steve and 4th cell contains my approach and we can see that the 4th cell has significant improvement in terms of outlier rejections and noise removal.

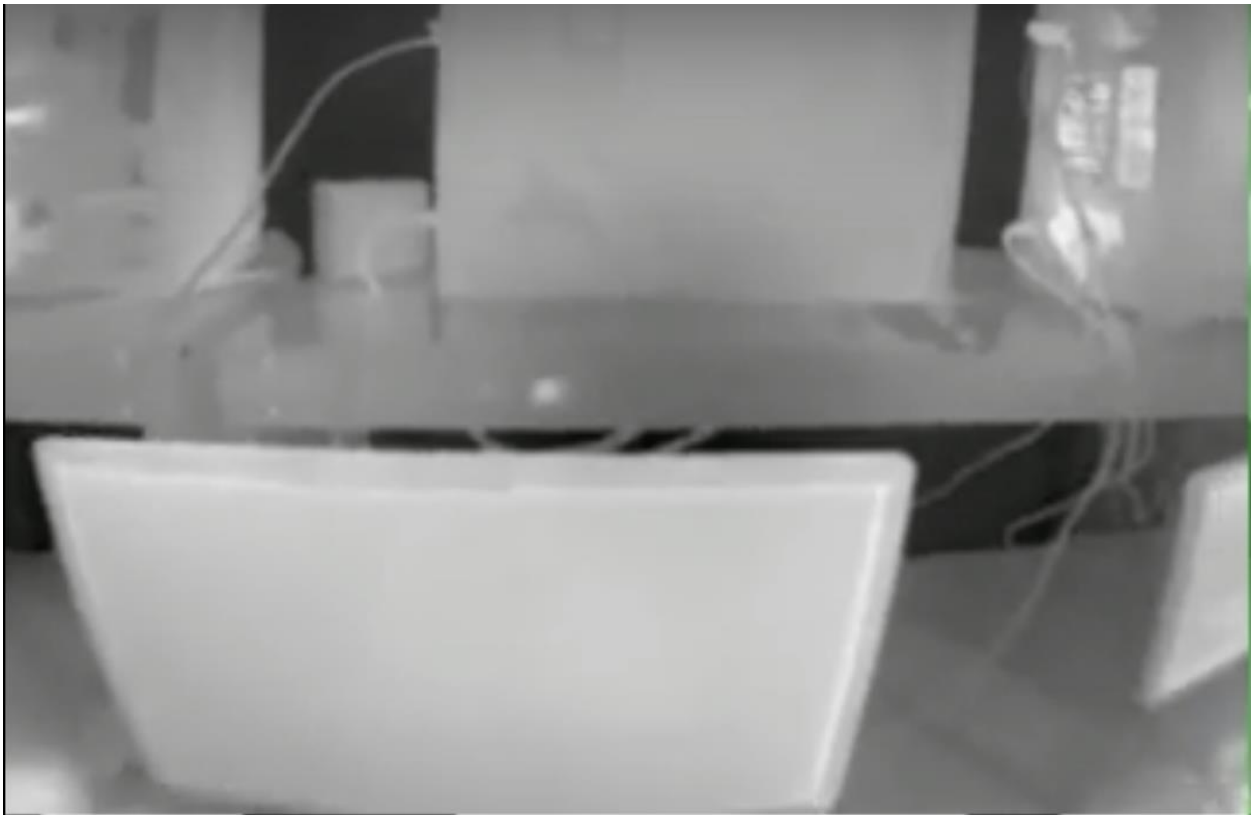

Figure 3: Input thermal image

Figure 4: Segmented output

As from the above images we can see that the hot desktop screen was detected successfully along with a outlier which depicts the hot CPU of the other computer as seen from the input image.

Challenges:

Since no one in the team had ever worked with a thermal camera before, I had to go over the available online resources such as the datasheet to actually understand how the camera works and how the images are formed, using this foundation the challenge was to get the images from the camera as there was no available driver for it, I struggled in tuning the available drivers but it did not work but luckily I was told by my sponsors about a student who is working on the same camera and has a driver for it so I used his driver and the issue got sorted.

It took me some time to understand why reflections from humans or from other high energy sources and I couldn't get rid of those as my algorithm thought that they were not outliers but eventually by applying a threshold and area approximation I was able to remove those outliers. Also it is very tricky to get around with lights as they show up as fire in the current algorithm and we don't want that but I have learned that there is a low sensitivity mode in the camera that helps me with images that are low contrast as slight temperature differences don't affect the corresponding pixel intensities much as compared to the high sensitivity mode where it's difficult to segment. Currently I don't know a way to put the camera in low sensitivity mode right from the time when it boots up, so I am trying to figure out a way to do this.

Also we have issues with bright objects which we don't want to classify as fire such as a mirror that reflects sun (MBZ is in outdoor environment and reflectance of sun will be a big challenge to handle) so we are planning to use a RBG camera with the thermal and use the RBG intensities as a mask for the current thermal pipeline which will help us eliminate the bright energy sources as they will have a higher intensity in the RGB image compared to the thermal image.

Teamwork:

The team is split into two smaller teams where I and Zhihao (Steve) were working on the thermal camera where Steve worked on the watershed algorithm and his custom OTSU implementation. Parv and Shubham were working on getting the ORB-SLAM algorithm on the ground and aerial vehicle and they faced issues with getting images from the ZED camera where I came in because I had worked with those cameras for the April Tag detection, so I helped them out. Apart from getting the SLAM running they also tested it out by creating a point cloud and verified the results by moving the camera.

Future Plans:

The future plans for the next presentation are to work on the AGV arm mounting and design (assigned to Parv). Steve will be interfacing the microcontroller with the jetson and Zotac. Shubham and Parv are responsible for demonstrating the SLAM integration in the state machine where Shubham will also work on the PCB for our Husky. Since from the videos we showed during the presentation we can saw that the drone is not rock-solid stable in outdoor environment and thus I will be carrying out the outdoor stabilization tests, gain tuning and will also take the pilots test on 03/09/2019.