



SE 101 Introduction to Methods of Software Engineering

Fall 2019

Course Project Final Report

Group Name: Brett Bois

Team Member	Student ID	Username
Jason Guo	20832323	j279guo
Jonathan Xu	20824466	j462xu
Joshua Cheng	20824494	j885chen
Ricky Mao	20816518	r23mao
William Qin	20828843	w29qin

1. Introduction

Due to climate change and human negligence, forest fires are becoming alarmingly common and frequent throughout the world. In Canada alone, we have spent around 500 million to a billion dollars each year to prevent and fight forest fires. Not only is it a burden on the government's budget, but on people who have been affected. Forest fires have cost affected communities up to 10 billion dollars in damages in recent records. Coupled with the fact that forest fires are becoming more intense, we are facing a very dangerous future.

To help with this major problem, we sought to provide another way to gather more information about forest fires, specifically image and temperature data. We hope that the data gathered onsite can help firefighters develop a better plan of attack and researchers better understand how forest fires start and proliferate for future fires.

Introducing *Project BRETT*, the autonomous heat source mapping drone. Built with fire resistant carbon fiber and powered by NXP electronics, *Project BRETT* comes fully equipped to support firefighters and researchers in the fight against fires. Collecting image and heat data is easy: all you need to do is tell the drone where to go, wait for it to come back, and run a program on the data to create a heatmap.

2. Background Research

I. Forest Fires

Information and statistics discussed in the introduction were from the Canadian government's forest fire statistics and articles. Additional information was found from reputable news sites and climate information sites (links in references).

II. NXP Hovergames Drone and Autopilot

Because of the nature of the problem we were trying to solve, our group wanted to use a programmable drone that was capable of data collection and autopilot. Moreover, we agreed that the focus of our project should be on the software applications of the autonomous drone, instead of the construction of the drone itself. As such we decided that the HoverGames drone kit offered by NXP was the best platform for the project. Throughout the project, we consulted the Hovergames gitbook site for documentation on assembling and flying (manual and autopilot) our drone.

III. Flight Management Unit Sensor Data Collection

In order to deploy custom code to the drone's flight management unit for collecting sensor data and writing to the microSD card we needed to install our custom firmware on the drone. Both the NXP Hovergames developer guide and the PX4 Development Guide were used to setup the developer environment and learn how to write proper code for the drone. In addition, PX4's GitHub example code was used for code snippets and to refer to needed header files and their proper implementation. Finally, the code was deployed using QGroundControl.

IV. Raspberry Pi Image Capture

With regards to image capture, we used a Raspberry Pi Zero W with a 5 megapixel PiCamera to get high quality photos in a small form factor. Since the Pi would be run headless and without any external SSH connection, the code segments from the "How to Make a Safe Shutdown Button for Raspberry Pi" tutorial offered examples for basic IO, headless shutoff, file IO, and interfacing with the PiCamera. This was very helpful in setting up the Pi Zero so that it could capture photos at regular intervals, and interfaced headless without ssh.

V. Python Infrared Heatmap Generator

The infrared map used data collected from a Melexis IR sensor on the drone. The data was collected and put on a text file that also contained information regarding the position of the drone in x, y, and z coordinates. Then we used Python to read the files given to us and transformed it into a dataset that we can plot with Matplotlib and Seaborn. In the process, we depended a lot on the documentation and tutorials on Seaborn and Matplotlib to create the heatmaps (see references).

VI. OpenCV Image Stitching Program

Taking into consideration the limitations of the image capture hardware, we looked for an image stitching technique that would work regardless of the orientation, size, or amount of overlap of the images. The article "Image Stitching with OpenCV and Python" offered insight into an algorithm proposed by the paper, "Automatic Panoramic Image Stitching with Invariant Features", which is insensitive to the orientation and ordering of the images. We found that the algorithm was already implemented in the OpenCV library in Python, which was discussed in detail in the library's documentation. The code segments presented in the first article were a great start to the image stitching program we ended up creating.

3. Implementation

See code on GitLab: <https://git.uwaterloo.ca/j885chen/BRETT-SE101>

I. NXP Hovergames Drone and Autopilot

To build the physical drone, we assembled the components provided with the HoverGames drone kit. First, the carbon fiber frame, landing gear, arms, and rails were assembled. The Power Distribution Board for the entire drone, and Electronic Speed Controllers for each brushless motor (2212 920KV) was then put together. We then attached the RC receiver module so that the drone could be controlled manually with a joystick RV controller, and added the NXP FMUK66 Flight Management Unit, powered by an external FMU power module. Finally, we attached a 3 cell 11.1v 5000mah lipo battery, ReadytoSky GPS module, Melexis IR sensor, and a set of 915mHz SiK telemetry radios so that the drone could communicate with a laptop running QGroundControl.

To program the drone, we first created a PX4 profile for the drone on QGroundControl, so that the software knows the dimensions, weight, and sensor readings of the drone when conducting autopilot flight. To ensure a degree of safety, we also set up a kill switch on the manual controller, and set the fail-safe features to account for situations when the drone runs low on battery, breaches the maximum allowable height, or loses its telemetry connection. Finally, we uploaded the provided bootloader to the FMUK66, and uploaded open source PX4 autopilot firmware using QGroundControl.

II. Flight Management Unit Sensor Data Collection

The sensor data collection program on the flight management unit was written in C++, using PX4 header files. First, the hg_temp.cpp class was written to get temperature data from the sensor. Then, the module.cpp class was written to create the main program that would run in the background. The program would poll for drone position data, and every 5 seconds if data was received, grab temperature data and write to a text file saved on the microSD. The program could be started and stopped at any moment, and the user could pass a parameter [-n] to indicate where they wanted the data to be saved.

Significant challenges were encountered in deploying the code to the drone, as well as writing to the text file without crashing the program. Initially, deploying custom firmware would break the drone's ability to fly. This was remedied by following the PX4 guide instead of the NXP Hovergames guide and reinstalling everything natively on Mac OS X. Program crashes due to writing to a text file were partly remedied by using fputs() instead of fprintf().

III. Raspberry Pi Image Capture

To capture images at regular intervals, without intervention from an external controller, the Raspberry Pi Zero W runs python code that incorporates the picamera and rpi.GPIO python libraries. Since the project will be run in an external environment without a debugging console, the script captures pictures only when GPIO input on pin 16 is high. When capturing begins, the Pi creates a new directory on the SD card, so that images recorded in the same flight can be grouped together. The Pi captures a photo every 3 seconds and records the image name and timestamp on a separate .txt file, so that the image stitching algorithm can reconstruct the full image based on the location of the drone at each timestamp. A significant challenge we faced was ensuring the camera connection is properly mounted, as the cable often disengaged during flight.

IV. Python Infrared Heatmap Generator

The creation of the infrared heatmap and flight path map was split into three main steps. Firstly, the program had to read in the text file sent to it, which contained information about the drone's position, as well as ambient and object temperatures. This was done with Python's built in file in functions. The second task was to parse that information to create a data frame compatible with our graph creating libraries. For the heatmap, it was sufficient to put it into a list, while for the flight path map, the helper function Pandas was used to create and pivot the information. Finally, the formatted data frame was used in conjunction with Seaborn and Matplotlib, two libraries used for data visualization, to create the respective visuals.

V. OpenCV Image Stitching Program

The image stitching program was written in Python, primarily using the OpenCV library. We used the "createStitcher" and "stitch" methods to combine the images, as well as some helper libraries such as numpy and imutils to make image parsing easier. To combine the heatmap and stitched images, we used the "addWeighted" method from OpenCV to blend the images together.

4. Group Members' Contribution

I. Jason Guo

I worked primarily on the image processing programs that stitched the drone's raw images together and merged the heatmap with the final image. I researched different image stitching techniques and chose the algorithm that would best suit our project.

II. Jonathan Xu

For this project, I assembled the materials and built the hardware aspect of the drone. I helped test the autopilot features of the drone, and uploaded the initial bootloader and flight control to the drone. I also worked on the image capture script for the Raspberry Pi.

III. Joshua Cheng

My role in the project was to work on the heatmap, by writing Python code to parse the data sent to me by the drone and create a heatmap out of it. This task was accomplished through researching python data visualization libraries and implementing them. I also helped with creating a map of the drone flight path.

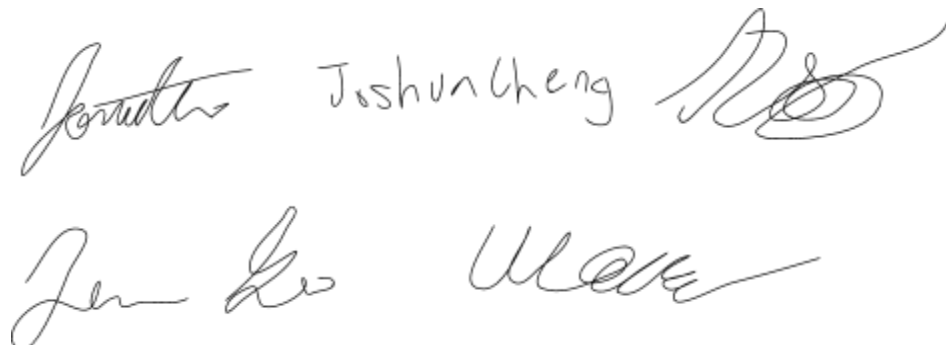
IV. Ricky Mao

I helped research and develop the heatmap generator and created the demo video for our project.

V. William Qin

I worked on deploying custom firmware and code to the drone's flight management unit, creating an executable program that could run from QGroundControl and would gather temperature data. I also helped with getting our drone to fly autonomously.

Signatures: (Left to right: Jonathan, Joshua, Ricky, Jason, William)

The image shows five handwritten signatures in black ink, arranged in two rows. The top row contains three signatures: 'Jonathan Xu', 'Joshua Cheng', and 'Ricky Mao'. The bottom row contains two signatures: 'Jason Guo' and 'William Qin'. The signatures are written in a cursive, flowing style.

5. Final Product Evaluation

I. Overall Evaluation

The original goal was to create an accurate heat map using an autonomous drone. Though we had to reduce the reliability and usability of our project by a bit, we successfully achieved our goal of creating an accurate heat map, as seen in our demo video. We are very proud of what we achieved and find that the performance of the final product has met our expectations.

II. NXP Hovergames Drone

The drone was assembled without major issues, and the drone performed quite well in flight. All the hardware features turned out to be fully operational. One slight issue was the weight balance of the drone, as its center of gravity is slightly offset, resulting in the drone drifting when in manual flight mode. The bootloader and PX4 firmware also worked as desired.

III. Flight Management Unit Sensor Data Collection

Data collection was sufficient but still unstable. The background program would sometimes crash the drone and sometimes work completely fine. This meant that we could still test the drone but we had to carry the risk of the program crashing the drone mid flight and having the drone literally crash into the ground. Even though the probability of crashes was minimized by reducing the frequency of writes to the microSD card, this problem made our program unreliable. However, the program still accomplished its goal and the data we received was exactly what we wanted.

IV. Raspberry Pi Image Capture

Initially, we hoped to capture images using the provided Pixy2 camera. However, we realized that the camera was designed for feature tracking and detection, and actually has an unsatisfactory resolution for capturing raw images. Moreover, the camera is only interfaceable with the FMUK66 via the CAN bus, however it can only send basic feature info instead of the full image data. As a result, our group decided to use a Picamera with an external Raspberry Pi Zero W. After the pivot, the image quality and interfacing went quite well.

V. Python Infrared Heatmap Generator

The program created to generate heatmaps and flight paths ended up being very successful. This was for two main reasons. Firstly, it met our primary objective of outlining large sources of heat, which was confirmed by testing it with a campfire. Secondly, the map was very accurate in terms of location and temperature, something of utmost importance to firefighters.

VI. OpenCV Image Stitching Program

We were pleased with the final result of the image stitching program. Originally, we thought that we would have to manually orient and filter the raw images taken by the drone; however, the algorithm we chose ended up taking care of this for us. The image processing programs exceeded our initial goals, and turned out better than we originally expected.

6. Design Trade-offs

I. Switching built-in PixyCam with Raspberry Pi + RPI Camera

A major design trade-off we had to make was to switch from the proposed PixyCam2 camera to a 5MP Raspberry Pi camera. The main reason for this switch was due to the fact that the PixyCam specialized in object tracking and feature detection, whereas our project simply needed high quality images to stitch together. In addition, the implementation for capturing raw pictures from the camera was very difficult, as it only supported picture transfers on its micro-usb UART bus, instead of the CAN interface we used to connect it to the FMUK66 flight management unit. As a result, we decided to mount a Raspberry Pi Zero W, powered by its own mini battery pack on to the drone, and synchronize the data collection between the FMUK66 and Pi Zero W. We would then read the stored log files and data from each board's corresponding micro SD card. This allowed us to easily capture high quality still images using the PiCamera library.

II. Interpolating temperature data using kernel density estimation

Due to the lower than expected number of data points collected by the drone, what was found was that parts of the heatmap lacked any data at all. This unforeseen circumstance led to a design tradeoff, which was to instead create the heatmap using kernel density estimation. What this meant was to plot the points collected by the drone, and then with an algorithm contained in the Python library Seaborn, estimate the temperatures to fill in missing data. Doing this allowed us to have a complete heatmap, with heat levels that propagated rather than cut off abruptly. Overall, this tradeoff was made due to hardware limitations, as it is unrealistic to get data of every single position.

7. Future Work

I. More Extensive Testing

In the interest of time, we were only able to test out our project in a contrived setting, with only one major heat source. In future work, we would like to perform tests in a more realistic setting with multiple large sources of heat. This would give us insights into how we can improve our heatmap generation to be of more use to researchers and fighters.

II. Improve User Interface

Currently, our project is not user-friendly and requires us to manually run code. We would like to improve the interface of our application to appeal to people with no technical background. Creating an app that allows users to control the drone's flight and image capture, while running image stitching and heatmap creation on the backend, would be a pragmatic improvement to our current design.

III. Upgrade Hardware

The current infrared sensors and PiCamera mounted on the Raspberry Pi proved to be limited in their capability to take accurate, usable data. Upgrading to higher quality hardware would improve the quality and level of detail present in the produced heat maps, as we would be able to take a larger volume of more sensitive temperature readings. In addition, higher resolution images would improve the clarity of the produced heatmap and would allow users to visually pinpoint the sources of heat.

8. References

- Choosing color palettes¶. (n.d.). Retrieved from https://seaborn.pydata.org/tutorial/color_palettes.html.
- Climate change and fire. (2019, June 18). Retrieved from <https://www.nrcan.gc.ca/our-natural-resources/forests-forestry/wildland-fires-insects-disturban/climate-change-fire/13155>.
- Cost of wildland fire protection. (2019, June 17). Retrieved from <https://www.nrcan.gc.ca/climate-change/impacts-adaptations/climate-change-impacts-for-ests/forest-change-indicators/cost-fire-protection/17783>.
- Das, A., Ebrahim, M., George, & Arun. (2019, March 26). Seaborn heatmap tutorial (Python Data Visualization). Retrieved from <https://likegeeks.com/seaborn-heatmap-tutorial/>.
- Forest fires. (2019, June 28). Retrieved from <https://www.nrcan.gc.ca/our-natural-resources/forests-forestry/wildland-fires-insects-disturban/forest-fires/13143>.
- Forest Fires and Climate Change. (n.d.). Retrieved from <https://climateatlas.ca/forest-fires-and-climate-change>.
- Getting started. (n.d.). Retrieved from <https://nxp.gitbook.io/hovergames/developerguide/>.
- Images stitching. (n.d.). Retrieved from https://docs.opencv.org/4.1.2/d1/d46/group_stitching.html.
- Introduction. (n.d.). Retrieved from <https://dev.px4.io/master/en/>.
- Kernel density estimation. (2019, November 27). Retrieved from https://en.wikipedia.org/wiki/Kernel_density_estimation.
- Maher, S. (2019, July 18). Climate change is making wildfires in Canada bigger, hotter and more dangerous. Retrieved from <https://www.macleans.ca/news/canada/climate-change-is-making-wildfires-in-canada-hotter-and-more-dangerous/>.
- PX4. (2019, August 14). PX4/Firmware. Retrieved from <https://github.com/PX4/Firmware>.
- QGC - QGroundControl - Drone Control. (n.d.). Retrieved from <http://qgroundcontrol.com/>.
- Rosebrock, A. (2019, May 15). Image Stitching with OpenCV and Python. Retrieved from <https://www.pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/>.

Rosebrock, A. (2019, March 18). OpenCV panorama stitching. Retrieved from <https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/>.

seaborn.heatmap¶. (n.d.). Retrieved from <https://seaborn.pydata.org/generated/seaborn.heatmap.html>.

seaborn.kdeplot. (n.d.). Retrieved from <https://seaborn.pydata.org/generated/seaborn.kdeplot.html>.