UPPSALA
UNIVERSITET

TVE19022
Examensarbete 15 hp
Juni 2019

Thermal Imaging Platform for Drones

Cost-effective localization of forest fires

Joel Bjervig
Johan Slagbrand

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Thermal Imaging Platform for Drones

Joel Bjervig, Johan Slagbrand

A device for identifying forest fires in an early stage has been developed during the course of this project. Attached to a drone, this prototype will provide a live-stream to a web server displaying a blended frame, made of a thermographic image showing thermal radiation and a regular photography with the visible light. The platform consists of a small single-boarded computer, a thermal camera sensor and a regular camera module. All powered by a power bank and fitted into a custom made 3D printed plastic case. At startup the computer automatically executes scripts written in Python, initializing its sensor components and processes the captured images which finally gets transmitted to a live-stream via a web server connection. Everything described above worked well, but originally the intent was for the web interface to provide a map with the current location coordinates of the drone. Since a module for mobile communication with support for GPS was not acquired, any implementation of such kind was impossible. However, several drone models already possess the feature to obtain such coordinates.

Handledare: Uwe Zimmerman, Jörgen Olsson
Ämnesgranskare: Jonas Lindh
Examinator: Martin Sjödin
ISSN: 1401-5757, TVE19022

Populärvetenskaplig sammanfattning

De senaste sommarperioderna har präglats av allt varmare väder och påtaglig torka. Som följd har antalet skogsbränder ökat runt om i landet, vilket förmodligen kommer bli vanligare de kommande sommrarna. Att hitta skogsbränder med hjälp av traditionella luftfarkoster så som helikoptrar och flygplan är mycket resurskrävande och ekonomiskt kostsamt. Utbildade piloter krävs för att manövrera dessa fordon som i sig är en dyr investering och även kostsamma i drift. I mån om att bidra till utvecklingen utav kostnadseffektiva verktyg som kan förebygga skogsbränder, har en lågbudgetlösning utvecklats för att upptäcka värmesignaturer från eldsvådor, innan de utvecklas till större skogsbränder. Denna plattform är tänkt att fästas på en drönare och kontinuerligt ta kort på landskapet som sedan skickas till en websida på internet. Varje enskild bildruta ska komma från två olika slags kameror; en vanlig som upptar det synliga ljus vi ser till vardags, och en värmekamera som uppfattar infrarött ljus, även känt som värmestrålning. Därav kan användaren tydligt se både värmekällan från värmekameran samt i mer detalj terrängen omkring källan genom den vanliga kameramodulen.

Contents

1 Introduction

Locating forest fires using aerial vehicles such as helicopters and airplanes are resource costly. Trained pilots must maneuver the vehicles, which in turn are expensive investments and operation costly. This project aims to examine the possibilities of a low-cost thermal imaging platform for detecting early stages of a forest fire which is characterized by it's heat signatures. Such a platform will be attached to a drone and will provide visuals in the infrared and visible light spectrum together with location coordinates of the drone. All imagery and information will be displayed on a web page on the Internet through mobile communication. Such a product would be able to supervise firefighters and rescue teams in deciding where to engage their means of personnel. Furthermore, a thermal camera is capable of "seeing through" smoke, making it possible to track people working with extinguishing the fire. Similar products already exist as package solutions fitted for drones, with a price in the five figure (SEK) range. Due to a limited budget, which is compatible with the low-cost development aspect of this project, a low-end infrared array sensor together with a camera module will be implemented onto a single board computer. While it is applicable for human search, focus lies mainly on detecting forest fires since they emit greater amounts of thermal radiation, thus being easier to detect compared to humans.

2 Theory

2.1 Raspberry Pi

The Raspberry Pi, Model 3 B is a small single-boarded computer commonly used for educational purposes and as a development tool. The operating system is the Linux distributed software Raspbian specially fitted for Raspberry Pi. This computer serves as the main processing unit and was regarded suitable due to its processor performance, specially to manage a rapid processing of captured frames and uphold the stream to a server. Additionally, it has features such as WIFI and ready third party modules that are easy to implement.

2.2 Infrared and visible light radiation

The wavelength of visible light occupies a small interval of 0.38 μm to 0.74 μm, in the electromagnetic spectrum in comparison to infrared (IR) radiation that spans a greater interval, namely from 700 μm to 1000 μm, which can be categorized further into: Near Infrared (NIR), Short Wave Infrared (SWIR), Mid Wave Infrared (MWIR), Long Wave Infrared (LWIR) and lastly Far Infrared (FIR), see figure 1.[1]

2.3 Thermal sensors

Thermal radiation emitted from surfaces due to theirs temperature can be detected by an IR-camera. The thermographic effective range of IR detection usually lies withing the MWIR and LWIR range. Devices for measuring MWIR radiation usually requires cryogenic cooling to function properly, making them big, expensive and hard to operate. That is why most thermographic sensors utilize the Long Wave Infrared part of the spectrum. These sensors can operate in ambient temperatures, making them substantially cheaper to manufacture contrary to those for detecting MWIR radiation. When incident electromagnetic radiation reaches a LWIR thermal camera, it is filtered out by a LWIR bandpass film of typically Germanium or chalcogenide glass. Only light within the LWIR range gets transmitted and heats up the microbolometric elements. Every such element has an electrical resistance dependent on the incident radiation it absorbs. By sensing these varying resistances, a temperature can be obtained.[1]

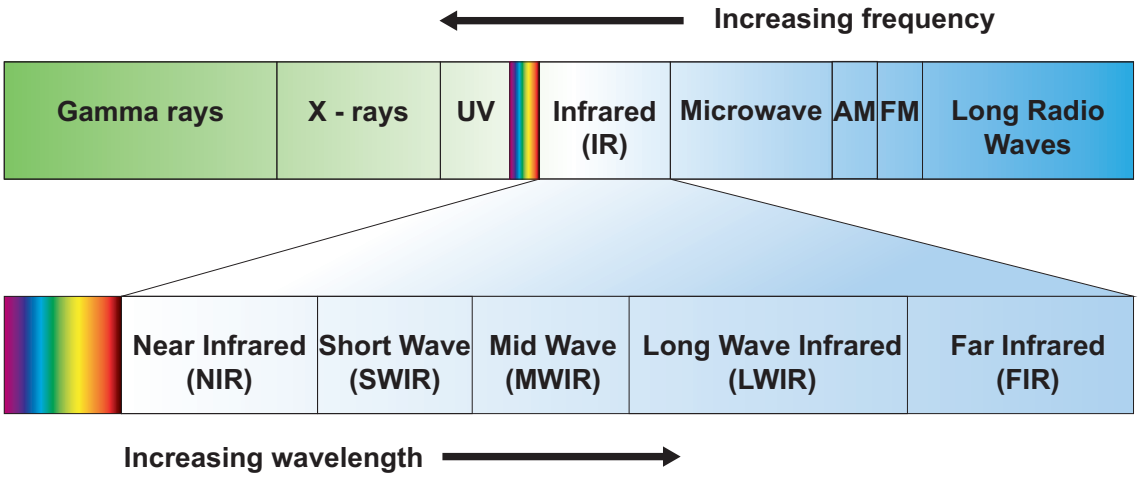

Figure 1: A detailed categorization of the infrared interval in the electromagnetic spectrum.

2.4 Raspberry Pi camera module

The Raspberry Pi camera module or picamera is a camera development kit from the Raspberry Pi Foundation. It possesses a Color CMOS sensor with a resolution of 8 Megapixels and outputs images of max 3280×2464 pixels (aspect ratio 4:3). Accompanied with a well documented Python library, the picamera is easy to control and supports various image formats.[2][3]

2.5 GIPO

GIPO, short for General Purpose Input/Output, is a type of pin allowing for easy access to the Raspberry Pi's system from the outside world. As the name states, it can act as either an output pin, sending out digital logic levels, such as controlling an led light or an electric motor. Acting as an input pin, reading digital logic signals from an external source.[4] The means of communications relies on binary threshold voltages. If output is set to HIGH or 1, the voltage over the pin will be raised 3.3 volt, while LOW or 0 equals to zero volt. Considering the input function, the signals must be more or equal to 3.3 volt to register as HIGH or 1, and zero as LOW or 0.[5]

2.6 Python

Python is a high level, general purpose programming language where the code is interpreted as it runs. Python follows an object oriented paradigm and is known to be easy to learn and therefore widely used. It comes with many well developed standard libraries. In addition there are numerous third parties libraries for example the picamera and the thermal camera AMG8833.

2.7 Image format

Several means of storing and organizing digital media exists and depending on the intentions and limitations, some formats possess more efficient algorithms than others, in terms of compression and bit rate. Developed in 1992 by the Joint Photographic Experts Group, JPEG is a still image format standard (ISO/IEC 10918) composed of several specifications and coding modes.[6]

2.8 RGB colour model

RGB stands for red, green and blue and they are the primary colours in the additive RGB colour model. The concept behind the model is letting the background be black and then add proportions between the primary elements red, green and blue (RGB) for generating any specific colour. A colour is represented by decimal code where each primary colour is assigned a value from 0 to 255 (256 values in total), which serves the purpose to fit into an 8 bit byte, since $2^8 = 256$. It is written on the form (R, G, B). For example black equals total absence of colour (0, 0, 0) and white corresponds to (255, 255, 255) which is full presence of colour. A natural comparison would be the

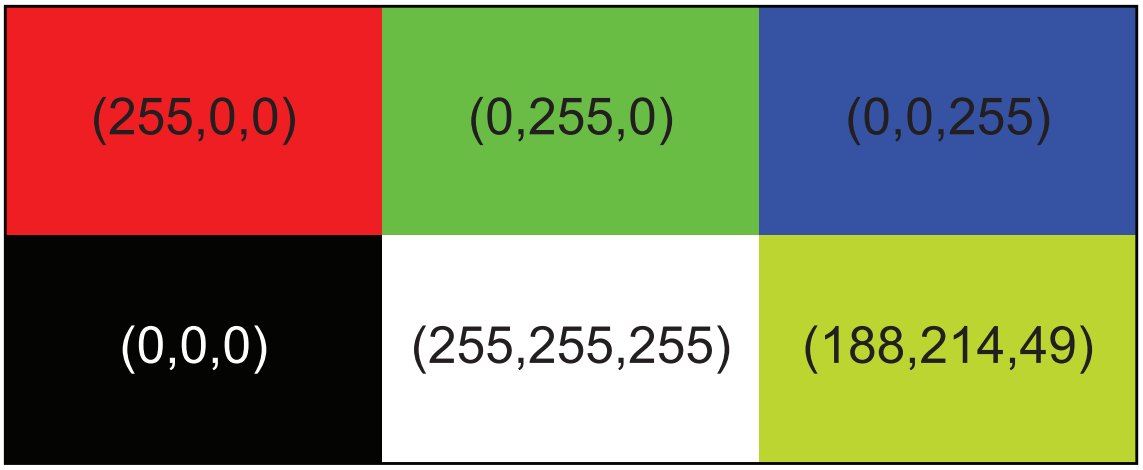

Figure 2: Caption

white light from the sun which consist of all the colours in the visible spectra. For demonstration, red is simply (255, 0, 0), shown in figure 2 together with some other samples. In many display monitors each pixel is defined by these three colour values and its placement on the screen.[7][8]

3 Method

3.1 Material

- Raspberry PI model 3B
- Panasonic Grid-EYE® Infrared Array Sensors
- Raspberry Pi camera module
- PocketCell powerbank, 6000mAh
- 3D-printed plastic case

3.2 Software development

Initially, the Raspberry Pi was acquainted with by connecting a screen, keyboard and mouse to it for exploring the environment in which all the software development where to take place. The code is written in Thonny, a Python Integrated Development Environment. The procedure behind the project was to first connect the picamera and install associated software. Next step was to connect the thermal camera sensor to the correct GIPO pins and install Adafruits library for utilizing the sensor. With all the components set the following milestone was to produce a heat image. Adafruits library contained an example script for reading the sensor and mapping the temperatures to colours, however the moving images it created couldn't be implemented. Therefore the code was rewritten to a format supporting image processing, mainly for fusing two frames together. When a final image could be obtained, it needed to be send to a server. To do so a script from GitHub written by Antonio ALVES,[9] was modified to suit the projects purpose.

3.3 Reading the sensors

The first step in the process was to gather data from the thermal camera. ADAFRUITS library ADA_8833 was utilised, which allows for easy reading of the sensor with the command *readPixels()*, generating an array containing temperatures in degree Celsius measured from the sensors separate elements. Concerning the picamera, the function command *picamera.capture()* generates an image with a specified output file format. In order to suit rapid processing and streaming purposes a lower resolution was set to 500×500 pixels.

3.4 Image processing

3.4.1 Temperature mapping

In order to visualise the thermal data in an informative and intuitive manner, the temperature values are mapped onto a colour gradient, ranging from blue to red with all the other colors in between, shown in Figure 3. Every time the sensor is being read, the lowest temperature is mapped to 0 (blue) and the highest temperature to 1023 (red). All the other temperatures in between are assigned correlative values within the interval. This is displayed in Figure 4, however the matrices are just for visualizing, in the actual code the sensor output is an 1×64 array, later resized to a matrix.

Figure 3: A colour gradient analogous to the one used to generate the thermal image.

22.0	21.5	22.0	22.5	22.0	22.0	21.75	21.5
22.25	22.25	22.75	24.75	22.5	22.0	21.75	21.5
23.25	25.5	24.0	25.5	23.5	22.0	22.0	22.25
26.25	27.0	23.25	23.75	25.0	24.25	21.75	22.0
27.5	27.0	27.25	25.25	25.0	26.0	23.25	22.5
27.5	26.25	25.0	27.0	25.5	25.25	24.0	23.5
25.5	25.0	23.5	28.25	26.75	25.0	24.25	23.5
24.25	23.75	24.75	24.75	25.0	24.25	24.5	23.5

76.0	0.0	76.0	152.0	76.0	76.0	38.0	0.0
114.0	114.0	189.0	493.0	152.0	76.0	38.0	0.0
265.0	606.0	379.0	606.0	303.0	76.0	76.0	114.0
720.0	834.0	265.0	341.0	530.0	417.0	38.0	76.0
909.0	834.0	871.0	568.0	530.0	682.0	265.0	152.0
909.0	720.0	530.0	834.0	606.0	568.0	379.0	303.0
606.0	530.0	303.0	1023.0	796.0	530.0	417.0	303.0
417.0	341.0	493.0	493.0	530.0	417.0	455.0	303.0

Figure 4: Mapping from temperature values to numbers in the interval 0-1023. In code this data is processed as arrays, but for demonstration purposes shown as matrices.

3.4.2 Interpolation

As previously stated, the resolution of the thermal sensor is fairly low, 8×8 pixels. Cubic interpolation made it possible to increase the resolution to 32×32, which results in a matrix $32^2/8^2 = 16$ times larger. Interpolation works by constructing new data points between a set of known points. The accuracy may differ depending on which method is used. However, details occurring in the set where a picture is taken isn't possibly to reproduce.[10] The original set of data is the one received after temperature mapping, shown to the right in Figure 4, the interpolated set of data is shown to the left in Figure 5.

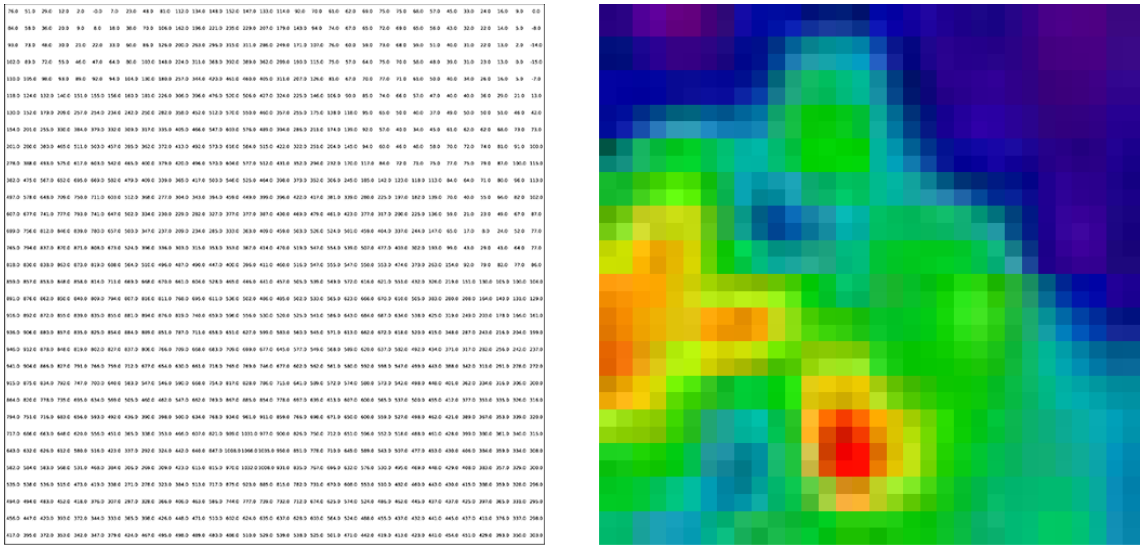

Figure 5: Visualizes the step from a 32×32 matrix with numerals to an image consisting of 32×32 pixels

3.4.3 Numbers to image

The values ranging from 0 to 1023 in 32×32 matrix are converted into decimal code in the RGB colour model. From the decimal code it is possible to generate the image shown to the right in Figure 5 with a function from a library named made by SciPy. Counting each colour element one can see the size is 32×32, however what is shown is an enlarged image of the original where the shape of the individual pixels have been conserved.

3.4.4 Resize with anti-aliasing

Next step is to resize the 32×32 image to a one with the size 500×500 in order to match the resolution of the picamera. Python Image Library (PIL) has an anti-aliasing filter when resizing images, meaning it will which "smooth out" the edges between the pixels when enlarged up, which is shown to the left in Figure 6.

3.4.5 Transparent image overlay

The camera and heat image are then blended into one final image with a pre-written function from PIL, adding them together with 50 % transparency on each. Worth to notice is when an image is fused from two sensor with a parallel distance between them, they won't completely overlap. Now the distance is fairly short and causes no problem as soon as objects aren't right next to the sensors. The final result is shown to the right in Figure 6, where image in the middle is the one captured by the picamera with text an overlaying text displaying the minimum and maximum temperature.

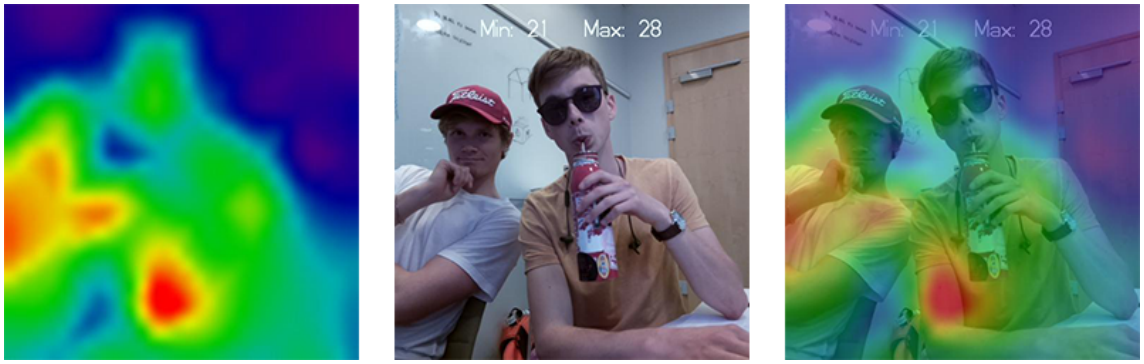

Figure 6: The heat image to the left is blended with the one in the middle from the picamera to produce the final frame to the right

3.5 Website

A local server is established in conjunction with running the scripts, ready to receive the stream of JPEG-images. To expose the local server to the Internet, making it accessibly for other units, a free service called Serveo is used. Serveo has the feature of creating your own sub-domain (depending on availability) and will provide a static URL to reach the local server.[11]

3.6 Startup routines

At startup the Raspberry Pi connects to a known network and automatically runs the script with the content described above. To enable that the scripts are located in an autostart file in a LXSession folder where self written startup routines are managed.

3.7 Case design

In the final stage of the project a 3D-printed case was manufactured to fit all the components to demonstrate how a complete platform could look like, shown in Figure 8, 9 and 10. It consists of five different parts, assembled with screws and mutters and the case can be attached to a drone or tripod by the built in camera mount. The idea behind the design of the case is to achieve a robust arrangement. Mainly since one frame from the stream is fused from the two sensor and therefore it's essential that they lay in the same plane without any angle between them. Another aspect in the design was to make room for ventilation slots to transfer away heat from the Raspberry Pi. The 3D-model of the case was designed in Autodesk Fusion 360, see Figure 7 and printed in a biodegradable plastic, polylactic acid (PLA). To strengthen up the cavities in the construction a cubic pattern fill was applied.

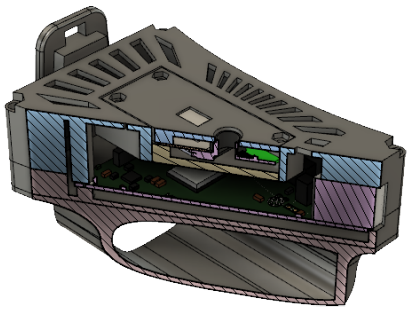

Figure 7: Cross sectional view of the case model in Autodesk Fusion 360

4 Results

4.1 Prototype

The result of this project is an apparatus attachable to a drone or a tripod which can provide a live-stream to a website showing a thermal view of the scenery together with the minimum and maximum temperature. At startup, that is when the Raspberry Pi is connected to a power source, it automatically executes scripts that utilize the sensor components, process each frame and establish a web server connection with a live-stream. In order to do so, it needs to be connected to a WIFI-network and presumably it could work with plugging in a 4G USB-modem as well. To access the website one has to type *mjao.serveo.net* in their address bar. The components are enclosed with a 3D-printed plastic case, allowing the user to access all ports to the Raspberry Pi.

Figure 8: The final product where all the components are fitted into a 3D-printed case.

Figure 9: A view from the back of the 3D-printed case, showing the power bank.

Figure 10: Showing the thermal camera sensor with the picamera above it.

4.2 Imagery from live stream

In figure 11 it is clear that the normal and the thermal image doesn't coincide completely. This is probably because the sensors are placed next to each other with some distance in between, and a small angle has occurred between the two planes in which the sensors lie, see Figure 10.

Figure 11: Showing a print screen of the live-stream on the web server. The minimum and maximum temperature are displayed in the upper part of the frame.

In Figure 12, the contribution of the thermographic sensor is clearly shown, in contrast to the normal image where no evident indication of a heat source is provided.

Figure 12: Shows a heat source from above with thermal perspective versus one without.

In Figure 13 the static and the adaptive method for colour visualization is displayed. At the left frame temperatures that are equal to or below 0 °C corresponds to blue and temperatures equal to or above 100 °C corresponds to red. At the right frame the lowest temperature corresponds to blue and the highest temperature corresponds to red. In the code one is given an option to chose between the two methods by manually comment out the static visualization.

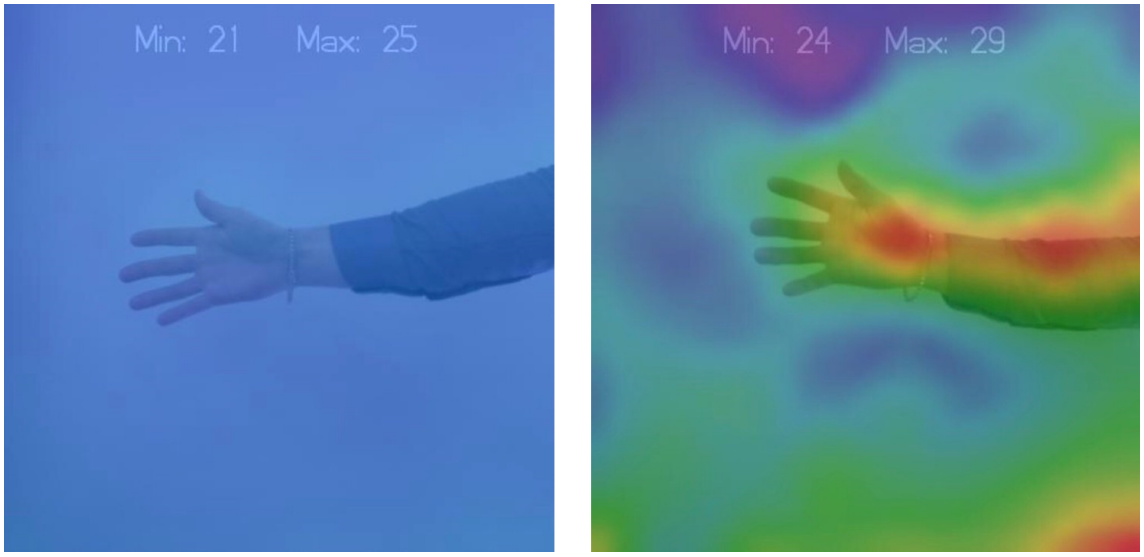

Figure 13: Images showing the static and adaptive colour visualization.

In figure 14, a sequence displays how the colour fluctuations increases with the distance from a heat source. One can also distinguish how the maximum temperature decreased for every frame further away from the grill. In the end of the sequence the heat source can no longer be detectable and thermal radiation from the ground closest to the camera dominates over the grill which causes the red spots in the bottom of the image.

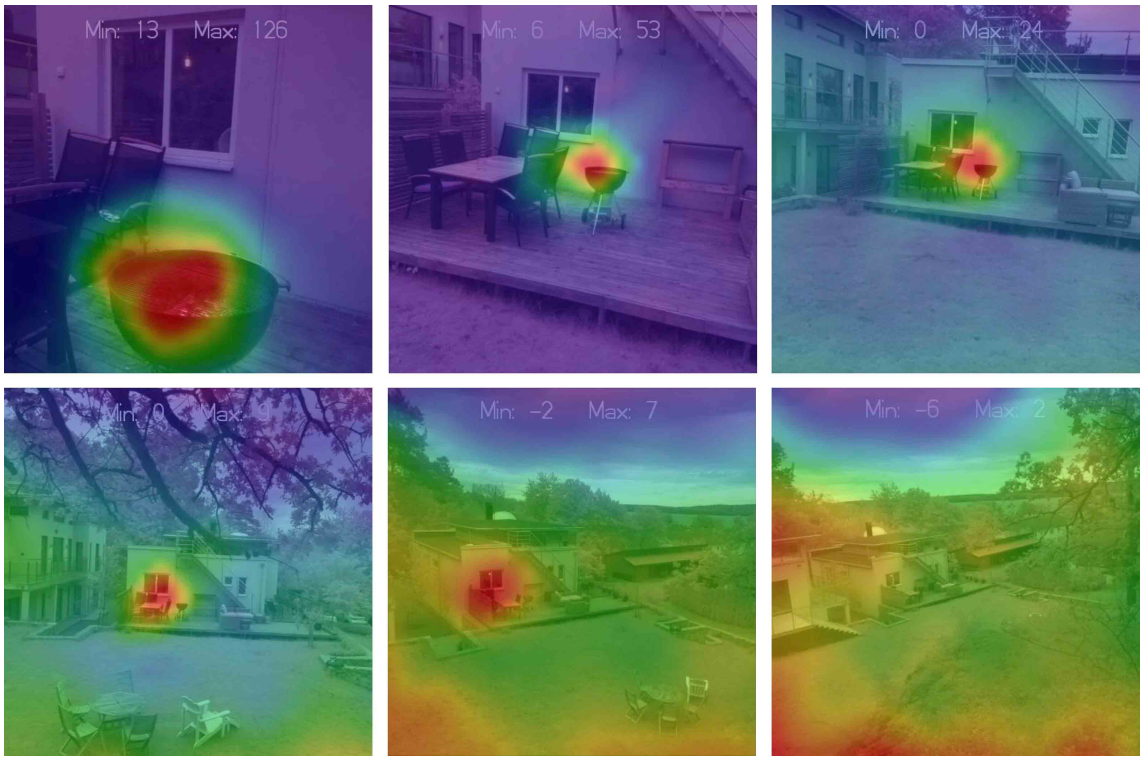

Figure 14: A sequence of frames with increasing distance from a BBQ grill, serving as a heat source.

5 Discussion

Over all, the project seems to have succeeded very well, even though one requirement that was specified in the planning stage regarding displaying a map with the coordinates of the position of the drone, couldn't be implemented, due to failure of acquire a module with mobile communication with GPS. The reason for that was because the component from the beginning was meant to be obtained from an old project at the university which had disappeared. Regarding the cost aspect, all components were provided by Uppsala University, free of charge, however the estimated total cost of the prototype is approximately 1000 to 1500 SEK depending on the retailers pricing. Implementing a GSM/GPS module, a lighter power supply arrangement and upgrading to a thermal camera with higher performance would increase the expenses further to about 4500 SEK. In full scale production though, all components would be purchased in greater quantities and consequently lower the price for each device.

5.1 Thermal camera

While Panasonic's thermal camera AMG8833 has a field of view considered sufficient, its resolution was not as high as high resolution thermography (i.e. thermal images rich in detail), which is necessary for long range thermal detection. FLIR offers the relatively low cost radiometric LWIR camera *FLIR Lepton 3.5®*. Its field of view with 57 degrees is slightly narrower than that of the AMG8833, which is compensated by its superior resolution of 160×120, making it 300 times finer than the AMG8833. Considering the price, Panasonic's sensor becomes more desirable. Sparkfun offers a complete kit with a breakout circuit the AMD8833 for roughly 400 SEK while FLIR offers

its LEPTON 3.5 for 2500 SEK and a breakout circuit for 2900 SEK. However, there exists cheaper and older versions of the Lepton module, complete with breakout circuits, such as the 2.5 version, available at a total of 2300 SEK.[12]

5.2 Frame rate

One area of improvement that doesn't necessary conflict with the application of such a device, is the rate of which the images get updated, i.e. the frame rate. The main issue that was faced was that the technique behind fusing the camera image with the heat image and some overlaying text relied on working with single images at a time. To do so a capture function from the picamera library was used that had a relatively slow encoder which was the main factor that limited the frame rate. In a potential next version, deeper research can be done in how to capture continuously images and match them with the frames generated by thermal camera sensor.

5.3 Adaptive colour visualization

One aspect when generating the thermal image was to consider pros and cons with static or adaptive temperature-to-colour mapping. Static means that a temperature is always represented by same colour which might be good in a situation when one wants to detect an object of any kind that completely stands out from its environment. An adaptive method on the other hand will visualize minor varieties in the thermal spectra when there are a small difference between the minimum and maximum temperature, for example a human in a room tempered setting. However the disadvantage with an adaptive algorithm occur when the environment has more or less the same temperature. The outcome will be a highly fluctuating stream, due to each frame contains red and blue fields appearing in an irregular pattern. For the purpose of detecting forest fires and plausible humans as well, an adaptive met would do fine, since the frequent fluctuations will diminish as soon as a significant temperature difference appear.

5.4 Case design

As seen in Figure 8, 9 and 10, no nuts or bolts are present in the assembly. They are temporarily replaced with cable ties (pink), since the right dimensions for some of the screws and mutters couldn't be acquired. In a previous version it was not taken into account that 3D printed models needs a sufficient margins in order for components to fit. To solve that a margin of 1 mm was added to all contact surfaces including the holes for the bolts and extrusions for the nuts.

5.5 Power supply

The weight of the entire platform as well as the size of the case is possible to reduce if using a lithium ion battery instead of the current power bank. Such a battery can't be connected directly to the GIPO pins, since they are lacking a built-in voltage regulator. Thus another module controlling a steady state power supply would be necessary which also will ensure that no power surge occurs that ultimately destroys the computer. A lithium ion battery together with a voltage regulator cost around 200-250 SEK. Furthermore those batteries can actually be dangerous if not cautiously handled, however keeping the platform light is essential if attaching it onto a drone. However, this improvement was discovered to late in the projects phase.

5.6 Code

The structure of the code may be improved by dividing some of sections into classes and methods, in particular the procedure of generating each heat frame. However a lot of inconvenience were encountered while attempting so. Features including iterative elements stopped working and it became difficult to gain a good overview of the code.

5.7 Additional sensors

In order to consider other sensor for the device, the properties of a wildfire must be analyzed. What is it that defines a fire from a human perspective? And which sort of electronic modules corresponds to these human senses? Humans can sense the radiation from a fire, not only in the visible spectrum

by our eyes, but also the radiating heat, warming our skin. These properties are well identified by the camera module and thermal camera. However, in many cases, the first impression of a forest fire is often the smell of something burning. Does there exist such a sensor that can "smell" smoke of burning wood? Smoke produced by forest fires contains over 90 % of water vapour and carbon dioxide (the other 10 % is of carbon monoxide, particulate matter, hydrocarbons and various organic compounds). Essentially a sensor detecting carbon dioxide and a humidity sensor could be implemented to the device, in order to detect the smoke from fire. Detecting an increase of carbon dioxide and humidity could both help ensure the presence of a fire. On the other hand, smoke production is oftentimes identifiable visually and whether these sensors really are necessary is questionable.[13]

6 Conclusions

Over all, the project seems to have succeeded well to judge from what has been achieved. However one requirement that was specified in the planning stage regarding displaying a map with the coordinates of the position of the drone, couldn't be implemented. The thermal camera functioned above the expectations. When testing the device on a BBQ grill, it registered heat radiation from approximately 30 m. The final cost for this prototype is estimated to be in the range of 1000 to 1500 SEK. If one wants to improve the platform so it could be used in a real scenario, more suitable components such as a different power supply setup, a thermal camera with higher performance and a GPS module will increase the price to around 4500 SEK. This low cost prototype shows that if developed further, it could very well assist firefighters in locating forest fires.

References

[1] AZO Sensors: An Engineer's Introduction to Thermal Imaging
https://www.azosensors.com/article.aspx?ArticleID=1157. Retrieved 20-05-2019

[2] Raspberry Pi: Camera
https://www.raspberrypi.org/documentation/hardware/camera/. Retrieved 15-04-2019

[3] Elinux: Rpi Camera Module
https://elinux.org/Rpi_Camera_ModuleTechnical_Parameters_.28v.2_board.29. Retrieved 15-04-2019

[4] Mosaic Documentation Web: GIPO Electrical Specifications
http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/raspberry-pi/gpio-pi
Retrieved 27-04-2019

[5] TechTerms: GIPO Defenition
https://techterms.com/definition/gpio. Retrieved 27-04-2019

[6] JPEG: Overview of JPEG
https://jpeg.org/jpeg/index.html. Retrieved 17-05-2019

[7] Charles A. Poynton (2003). Digital Video and HDTV: Algorithms and Interfaces. Morgan Kaufmann. ISBN 1-55860-792-7.

[8] Rapidtables: RGB color
https://www.rapidtables.com/web/color/RGB_Color.html. Retrieved 20-05-2019

[9] GitHUb: PiCameraStream
https://gist.github.com/nioto/9343730?fbclid=IwAR0fKY6URG33SJDcCfZw3gMaugP0cffTw4IsBwW3GIihC32u
Retrieved 19-04-2019

[10] Cambridge in colour: Image interpolation
https://www.cambridgeincolour.com/tutorials/image-interpolation.htm. Retrieved 20-05-2019

[11] Serveo: How it works manual and alternatives self-host
https://serveo.net/. Read 19-04-2019

[12] Mouser Electronics: Panasonic Grid-EYE ®Infrared Array Sensors
https://www.mouser.se/new/panasonic/panasonic-grid-eye-infrared-array-sensors/.
Retrieved 15-04-2019

[13] Auburn university: Smoke production
http://www.auburn.edu/academic/forestry_wildlife/fire/smoke_guide/smoke_-production.htm. Retrieved 20-05-2019

7 Appendix

```
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from Adafruit_AMG88xx import Adafruit_AMG88xx
from scipy.interpolate import griddata
from scipy.misc import toimage
from colour import Color
from PIL import Image
import numpy as np
import picamera
import math
import io

# set resolution for picamera
h = 500
w = 500

# initiate picamera object
cam = picamera.PiCamera()
cam.resolution = (w,h)

# set resolution for interpolated thermal image
height = 32
width = 32

# initialize the sensor and environment
sensor = Adafruit_AMG88xx(00, 0x68, None)

# how many colours we can have
COLORDEPTH = 1024

# set grid for interpolation
points = [(math.floor(ix / 8), (ix % 8)) for ix in range(0, 64)]
grid_x, grid_y = np.mgrid[0:7:32j, 0:7:32j]

# list of colours
blue = Color("indigo")
colors = list(blue.range_to(Color("red"), COLORDEPTH))

# create the array of colours
colors = [(int(c.red * 255), int(c.green * 255), int(c.blue * 255)) for c in colors]

# utility functions
def constrain(val, min_val, max_val):
    return min(max_val, max(min_val, val))

def map(x, in_min, in_max, out_min, out_max):
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# camera handler
class CamHandler(BaseHTTPRequestHandler):
  def do_GET(self):
    if self.path.endswith('.mjpg'):
      self.send_response(200)
      self.send_header('Content-type','multipart/x-mixed-replace; boundary=--jpgboundary')
      self.end_headers()
      stream=io.BytesIO()
      try:
        while True:
            # read the thermal camera sensor
            pixels = sensor.readPixels()
```

```
        # adaptive min and max temperature colour visualization
        MINTEMP = min(pixels)
        MAXTEMP = max(pixels)

        # overlay text with min and max temperature
        cam.annotate_text = 'Min: ' + str(int(MINTEMP)) + ' Max: ' + str(int(MAXTEMP))

        # picamera campture to stream
        cam.capture(stream, 'jpeg')
        camim = Image.open(stream)

        # static min and max temperature colour visualization
        #MINTEMP = 0
        #MAXTEMP = 80

        # map temperature
        pixels = [map(p, MINTEMP, MAXTEMP, 0, COLORDEPTH - 1) for p in pixels]

        # perform interpolation
        bicubic = griddata(points, pixels, (grid_x, grid_y), method='cubic')

        # create empty array
        heat_arr = np.empty((width, height, 3), np.uint16)
        heat_arr.shape = (height, width, 3)

        # convert to RGB decimal code
        for ix, row in enumerate(bicubic):
            for jx, pixel in enumerate(row):
                color = colors[constrain(int(pixel), 0, COLORDEPTH- 1)]
                heat_arr[ix,jx,:] = color

        # convert to image and resize with anti-aliasing
        heatim = toimage(heat_arr)
        heatim = heatim.resize((w,h),Image.ANTIALIAS)

        # convert to RGBA format
        background = heatim.convert("RGBA")
        overlay = camim.convert("RGBA")

        # overlay of the two images 50% transparency
        new_img = Image.blend(background, overlay, 0.5)

        # reset stream and save image
        stream=io.BytesIO()
        new_img.save(stream, 'jpeg')

        self.wfile.write("--jpgboundary")
        self.send_header('Content-type','image/jpeg')
        self.send_header('Content-length',len(stream.getvalue()))
        self.end_headers()
        self.wfile.write(stream.getvalue())
        self.end_headers()
        self.send_header('content-type','text/html'.encode())
        self.wfile.write(str(MAXTEMP).encode())
        self.end_headers()
        stream.seek(0)
        stream.truncate()
    except KeyboardInterrupt:
      pass
    return
  else:
    self.send_response(200)
    self.send_header('Content-type','text/html')
```

```
      self.end_headers()
      self.wfile.write("""<html><head></head><body>
        <h1> ThermalCam LIVE</h1>
        <img src="/cam.mjpg"/>
      </body></html>""")
      return

def main():
  try:
    server = HTTPServer(('',8000),CamHandler)
    print("server started")
    server.serve_forever()
  except KeyboardInterrupt:
    cam.close()
    server.socket.close()

if __name__ == '__main__':
  main()


import os

# expose local server to the Internet
os.system("ssh -R mjao:80:127.0.1.1:8000 serveo.net -N")
```