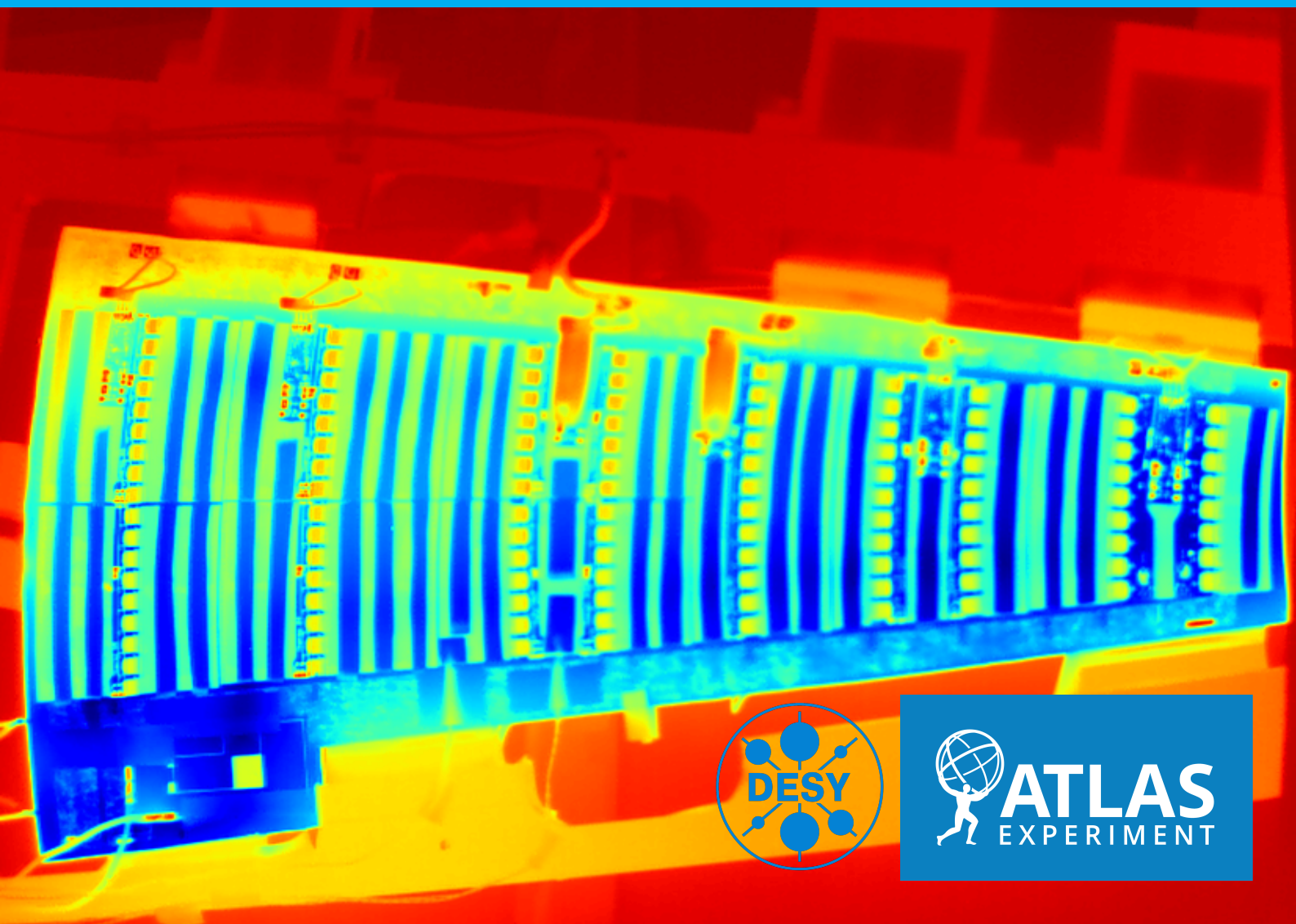
Thermal imaging of silicon detector components

Strip tracker endcap for ATLAS / HL-LHC

Maximilian Caspar



Thermal imaging of silicon detector components

Strip tracker endcap for ATLAS / HL-LHC

by

Maximilian Caspar

Final report on summer student project, DESY ATLAS group, Hamburg, 2017

Home university: University of Wuppertal

Project duration: July 18, 2017 – September 7, 2017

Special thanks:

Dr. Claire A. David	DESY, supervisor
Dr. Sergio Díez Cornell	DESY, supervisor
Yasiel Delabat	DESY, PhD student
Jan-Hendrik Arling	DESY, PhD student
Prof. Dr. Christian Zeitnitz	University of Wuppertal, referee
Prof. Dr. Karl-Heinz Kampert	University of Wuppertal, referee

An electronic version of this report is available at
http://www.desy.de/f/students/2017/reports.

Summary

The Deutsches Elektronen-Synchrotron (DESY) is currently working on the upgrade of the inner tracker end-cap of the ATLAS detector, of the Large Hadron Collider (LHC). The thermal properties of these end-caps have yet to be compared to simulations and a reliable method is needed for quality-control for withstanding the harsher environment of the High-Luminosity LHC program (HL-LHC). Infrared thermal imaging is used during the R&D phase, since it allows for contactless measurement of temperatures. When dealing with silicon however, certain corrections to the thermal image have to be made in order to create an accurate thermogram.

In this report, I will first explain the setup used for measuring the petal properties and go into detail about the changes I made to the instrumentation and data taking automation as part of my project. Then I will discuss the basics of infrared thermal imaging and explain the proposed corrections to our thermograms. At last I will present the results I have obtained using a calibration method focusing on known emissivity reference points.

Maximilian Caspar
Hamburg, September 2017

Contents

1 Introduction

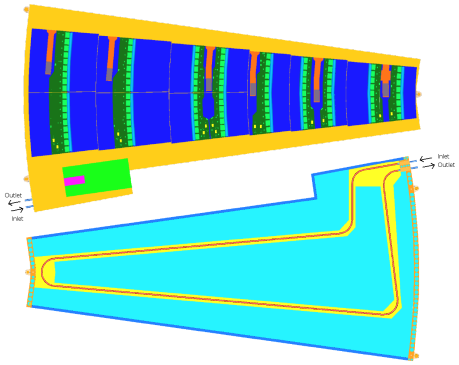
To increase the luminosity of the LHC by five to seven times its current value of $1.7 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, the biggest collider in the world will receive a major upgrade during its third long shutdown at around 2022-23. To cope with this huge amount of collisions, all experiments located at the LHC have to be upgraded as well. The increased radiation from the collision point requires higher radiation hardness of all components used in ATLAS, the high collision density in each event will require more resolution as well as faster readout electronics, which can deal with even bigger data rates.

1.1. The ATLAS strip detector

The current ATLAS SCT has around 6 million channels with an area of $60\,\mathrm{m}^2$, while the upgraded strips tracker will have 60 million of them, as well as a surface area of $160\,\mathrm{m}^2$. This almost four times denser detector will require more cooling than the old design.

The tracker will consist of one cylindrical section called ”barrel”, closed on each side with two disk lids referred to as ”end-caps”. While the module on the barrel parallel to the beam are mounted on long straight structures called staves, the barrel detector modules are mounted on petals shaped like disks (see [3] for more details).

Figure 1.1: A petal with its integrated cooling pipe



The petal The petal is a frame to mount the sensor endcap strip sensors to. It provides the mechanical and electronic structural support and also contains a titanium alloy cooling pipe for evaporative CO_2 cooling.

Since the modules are not produced yet I studied the petal prototype produced at DESY. It is equipped with the dummy electronics parts and mechanical components to be a thermal equivalent to the final structure, only the sensors are blank silicon dummies.

1.2. Basics of infrared thermal imaging

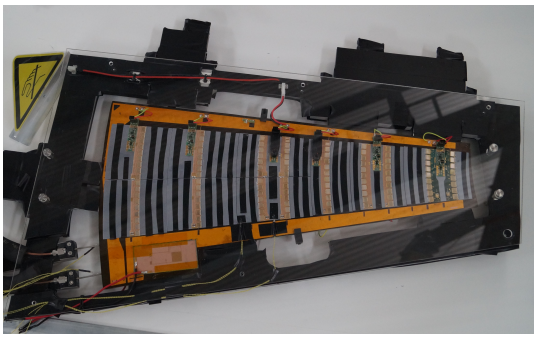
A perfect black body will emit a well defined spectrum of radiation, depending only on the temperature of the object. A thermal camera captures the intensity of the thermal radiation in the infrared (IR) band (our camera captures 7.5 µm to 14 µm). The temperature for each pixel can be calculated using Stefan–Boltzmann's law: $P = \sigma A T^4$. Since the camera is not probing the entire electromagnetic spectrum, a minor correction has to be made using the form of Planck's law.

Unfortunately, most real objects do not act like black bodies. Real bodies have an associated emissivity ϵ, reflectivity (R) and transmissivity (τ). ϵ, R and τ represent the fractions of power emitted, reflected and transmitted through the object, therefore $\epsilon + R + \tau = 1$. For more details on infrared thermal imaging, see Reference [6]

The problem with silicon is the irregular emissivity. Electronic structures on the wafer and other surface features will heavily influence emissivity. Additionally, the emissivity is not a simple material constant but a function of the wavelength [4].

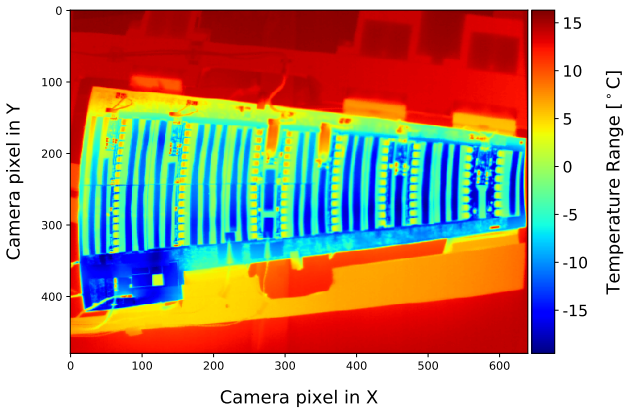
1.3. The black tape method

Figure 1.2: The petal prepared with stripes of black tape



In order to calibrate our silicon surface of unknown emissivity, we tried to establish a calibration profile using a spot of the same temperature and known emissivity. We used black tape with an emissivity of $\epsilon = 0.95$ to place stripes on the sensor surfaces. The idea is to measure the apparent temperature both on the black tape and on a point on the silicon directly next to the surface. Assuming both temperatures are actually the same, we want to obtain a relationship between the "accurate" black tape temperature and the measured silicon surface temperature.

Figure 1.3: A thermogram of a petal prepared with black tape. The difference in emissivity / temperature is clearly visible.



2

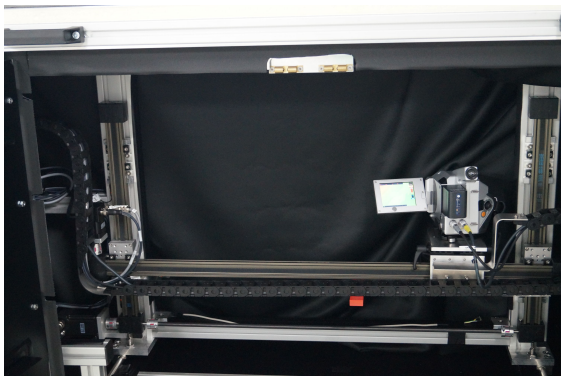
Experimental setup

2.1. The thermal chamber

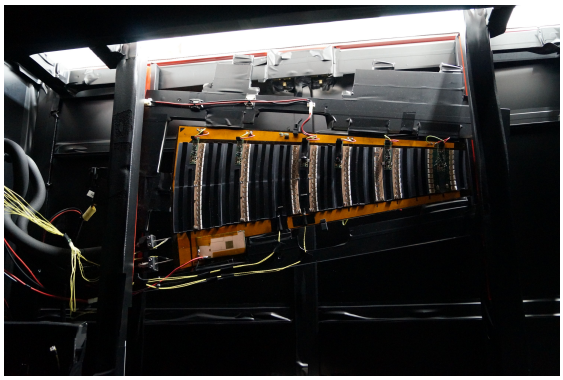
The main experimental area was inside an isolated chamber, where the petal was mounted. The camera was placed on a robotic gantry system on the opposite side of the chamber, infrared reflections from behind the camera were blocked using a black curtain.

Figure 2.1: The inside of the thermal chamber

(a) Curtain with IR camera and robotic gantry



(b) Petal mounted inside of the chamber



The petal cooling was provided by an evaporative CO_2 cooling system from CERN called TRACI (Transportable Refrigeration Apparatus for CO_2 Investigation). The TRACI is located outside of the chamber and connected with isolated cooling pipes. We used a vacuum pump to evacuate the air inside the petal cooling loop before filling it with CO_2.

2.2. Setup automation

Before my arrival, the measurements were done by reading the instruments' displays by hand. The instruments included a Keithley 2700 Multimeter, two TTI CPX400 power

Figure 2.2: Cooling and instrumentation

(a) TRACI (red box) and thermal chamber (white box) together with the vacuum pump.

(b) Additional instrumentation for the setup: power supplies and multimeter

supplies and a couple of thermocouples connected to a Raspberry Pi running a python script. Since reading out the instruments by hand is tedious and requires presence at the experiment (which was located at the DESY testbeam area), we decided to create a python script that provides an interface for the instruments and saves the data to a text file.

2.2.1. Device scripts

Keithley 2700

The multimeter was used to measure PT100 thermocouples using 4 wire sensing. Since the TRACI data readout was unworking we decided to directly connect its readouts (voltages) to the Keithley. It was connected to a computer using a RS-232 to USB adapter. We had to measure both voltages with 2 wire channels and temperatures with 4 wire sensing. The Keithley 2700 has two slots for relay cards in the back, we used both slots to retrieve the necessary signal from all channels.

The script is using the *pySerial* package in order to connect to the RS-232 cable. A *Keithley_2700* - class is created, this class handles the serial communication with the device, the switching between channels and the switching between temperature and voltage mode.

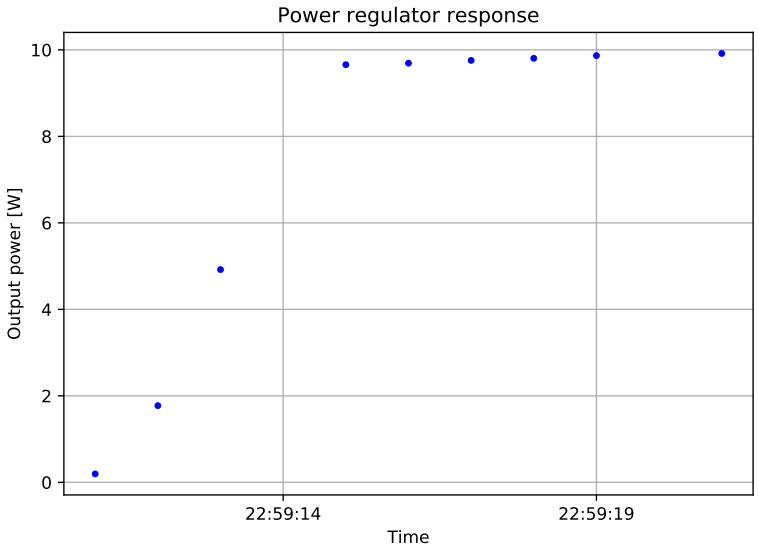
TTI CPX400

The petal is equipped with four dummy circuits simulating the heat emitted by the readout electronics. In order to ensure constant heating power, we used to set the power supplies by hand. This approach is a little bit tricky, because the resistance of the dummy modules actually changes when they heat up, changing the heating power at a fixed voltage. My idea was to interface both supplies to the computer via a serial cable and program a power regulator that also logs the power drawn by the modules.

The fast regulator behaviour was achieved using a multistate regulator algorithm. I regulate the output voltage by adding either 1 V, 0.1 V or 0.01 V to the output, depending on the distance of the current power from the goal. The behaviour while regulating down is equivalent.

I chose this technique because the output voltage can only be set in 0.01 V steps

Figure 2.3: Example behaviour for the power regulator (setpoint is 10 W)



anyway, making a PID algorithm unnecessary. When the difference between observed power and setpoint becomes smaller than a certain margin, the regulator does not add or subtract from the output voltage in order to reach a stable state.

Raspberry Pi

Additional thermocouples and humidity sensors are connected to a Raspberry Pi via an I^2C bus. The Raspberry Pi already had a readout script that writes the measured values plus a timestamp into a .csv file. The last line of this file would therefore be the current value. Every day, a new .csv file is created.

Via the terminal, this can be read out using `tail FILE.csv --line=1`

I then went on to write a python class that is capable of retrieving the values via ssh using the *Paramiko* package. First the file name of the latest .csv is determined by running `ls DIR/*.csv -rt | tail -1` and parsing the output. Then the script continously reads the latest data from that file using the same command that would be used when measuring manually via terminal.

2.2.2. The main measurement script

To connect all the instruments and have a convenient way of data taking, I wrote a main measurement script that loads all other instrument specific code and handles the file IO. There are several details of the script that serve an useful template for these kinds of programs in the future.

Buzzergen

One concern with data taking is timing. The script need to be sampled at discrete time intervals. When using a simple delay, one can expect to have the sampling rates slowly shift because of small differences in the timing of each cycle (e.g. because of delays in

the serial communication).

Jan Vlcinsky, a Stackoverflow power user, suggested a different approach in 2014, exploiting the python generator construct to create equidistant timing. The idea is to have a generator that only yields values when a timing goal is hit. The main measurement loop can just be implemented by `for i, t in buzzergen(DELAY):` and will always have correct timing.

Figure 2.4: The buzzergen routine as suggested in [5]

```
1  from itertools import count
2  import time
3
4  def buzzergen(period):
5    nexttime = time.time() + period
6    for i in count():
7      now = time.time()
8      tosleep = nexttime - now
9      if tosleep > 0:
10       time.sleep(tosleep)
11       nexttime += period
12     else:
13       nexttime = now + period
14     yield i, nexttime
```

Secure file IO

Storing data in text files is a fairly easy way of saving values, but to ensure with Python that a line written to a file ends up on the hard drive, a few extra steps have to be taken.

The normal way of writing files in python is to fill a array and write this array in the end of the program, or to use a `with open("NAME.txt")as f:` type of construct. One has to keep in mind that this kind of file stream is buffered. Therefore if the script crashes, there is no way to tell how much of the file has actually been written to hard drive.

In case of small data rates (like writing a line to a text file), directly writing to the hard drive is by far superior. One can rely on the last line in the file to be the most recent, even in the event of a crash. In practice, unbuffered writing can be implemented this way:

Figure 2.5: Use of the file IO concepts in combination with `buzzergen()`

```
1  f = open(fname, "w", 0) # The additional 0 ensures unbuffered writing
2  #Do the measurement setup
3  try:
4    for i in buzzergen(DELAY): #Measure and format a line
5      f.write(row + "\n") # f.write() does NOT generate a new line !!!
6    f.close()
7  except:
8    err = "Finishing off due to:" + str(sys.exc_info()[0]) #Get system error
9    print err
10   f.write(err + "\n")
11   f.close() # Close file
```

All these scripts have so far completed two thermal cycles over 24 hours and will therefore be used in the future experiments.

3

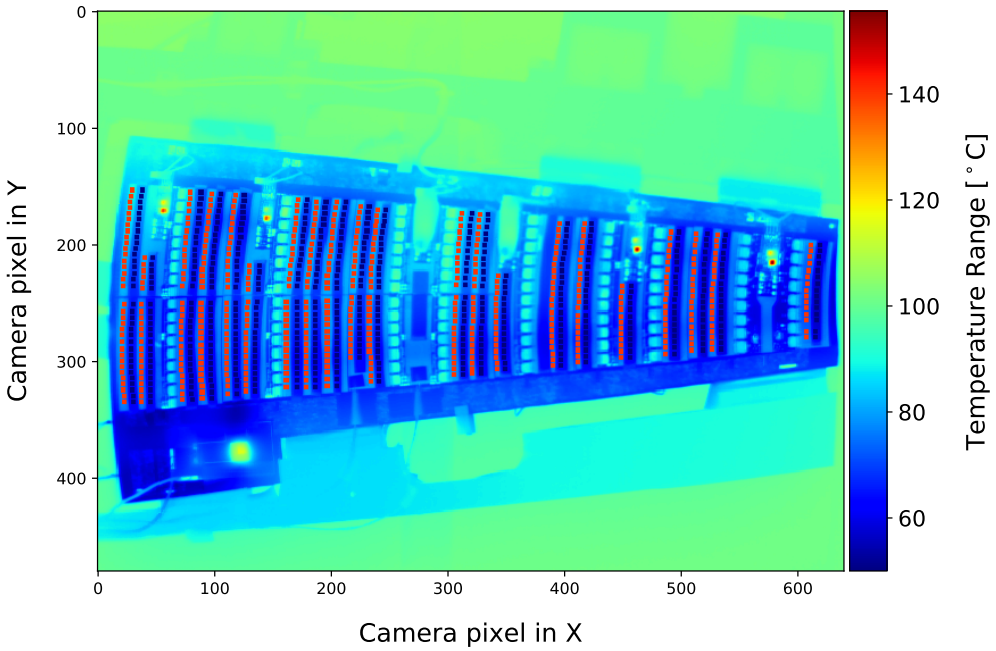
Data analysis

The main part of the data analysis was performed within ROOT, but several steps had to be taken inside python in order to convert the data to a useful format.

3.1. Markers

Markers are shapes that define an area of measurement within the thermogram. Since the thermogram is an image and can be represented as a two dimensional array of values, it seems reasonable to use rectangular markers.

Figure 3.1: Markers on a thermogram. The blue squares are associated with the black tape, the red squares are on the silicon surface



The first thing to do was to create a marker class, that has a rectangular shape associated to it. The markers are parsed from a text file that is created with the infrared camera software (IRBIS). After all markers are imported the marker can be called with an image array and returns the average value within the marker area as well as the standard deviation.

3.2. Thermograms

Thermograms can come in two different formats. They can either contain power or temperature data. They are provided by IRBIS as .asc files, which basically contain tables in the separated value format. I used the *numpy.genfromtxt* routine in order to read the data and convert all of the strings within the resulting array to floats.

IRBIS uses an image coordinate system that is flipped with respect to the coordinate system of the markers. In order to get the marker positions imported from IRBIS right with respect to the actual image, one has to switch the x and y axis of the data or the markers. In my macros, this is taken care of by the marker class.

3.3. Conversion to ROOT

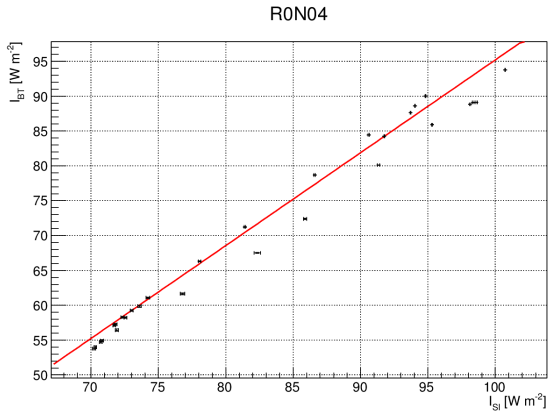
For data analysis purposes I decided to first save all the relevant data of the thermal cycle in one ROOT file [2] to make the data more accessible. Every thermal image was given its own TTree [1] filled with the marker names and the corresponding mean power and standard deviation for the silicon and the black tape marker.

Since looping over the trees in a ROOT file is not easy to do in python, I decided to save a tree called *T_NAME* containing the names of all other trees to make things easier.

3.4. Linear model

The first idea was to use a linear model fitting the relationship between the Intensity on the black tape and on the silicon for a given location on the petal. Using the ROOT file we created earlier, we are able to display the relationship between I_{Si} and I_{BT} and create a linear fit.

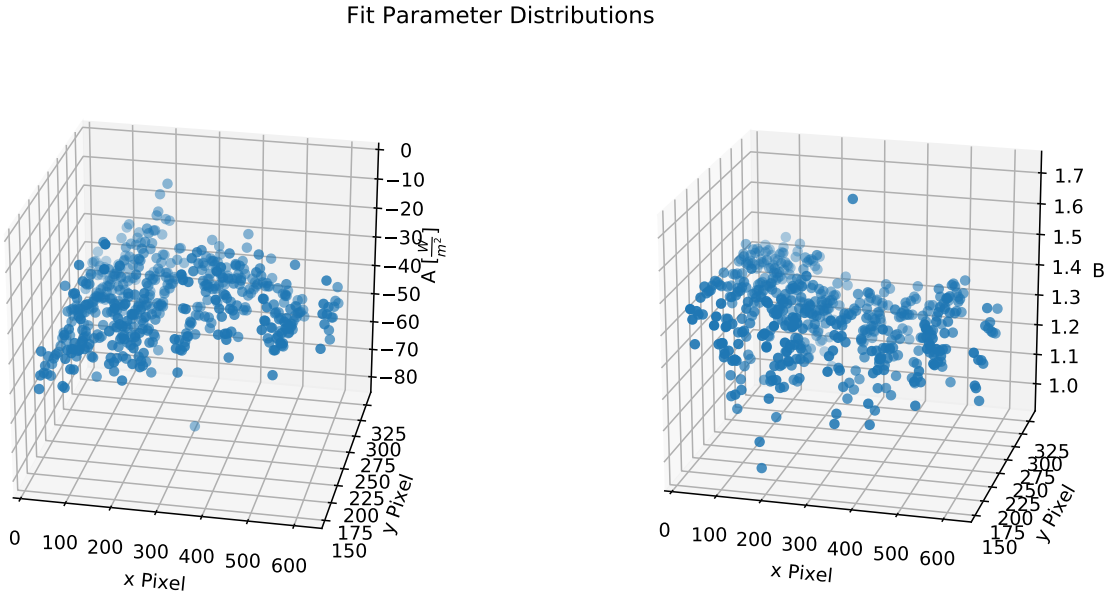
Figure 3.2: Data from a single marker in the thermal cycle with a linear fit



It seems like there are two overlaying effects. Since we have a thermal cycle, it is

possible that we are looking at a hysteresis effect. The electronics are another candidate for this split in the data. The next important step in the creation of a calibration map is investigating whether both lines can be separated using this information or if the split persists.

Figure 3.3: Fit parameters distributed over the petal surface. There does not seem to be a obvious relation between slope/offset and position. The fit was of the from $I_{\mathrm{BT}} = A + B \cdot I_{\mathrm{Si}}$ (A left, B right)



4

Conclusion

As part of the R&D for the new Inner Tracker (ITk) of the upgraded ATLAS detector at the LHC, two thermal cycles have been performed at DESY on the prototype end-cap petal.

The first challenge was the test setup and automation. I was able to get all devices in the experimental setup to work and managed to create a central script controlling the entire experiment. The amount of data accessible from the thermal cycles has gone up dramatically because of this, therefore more in-depth measurements can be performed in the future. This will make the study of the setup itself (temperature and humidity stability in the chamber, stability and resistance of the electronics, etc.) possible and will create more insight into the parameters of the thermal environment.

The data we got from the thermal cycles was successfully converted to the standard ROOT data format, enabling a more detailed analysis of the thermal cycles. The analysis done so far hints at two physical processes in the same thermal cycle, investigations in hysteresis or other parameters of the cycle might provide a explanation for the line separation we see in our data.

Bibliography

[1] TTree class reference, https://root.cern.ch/doc/master/classTTree.html.

[2] R. Brun, F. Rademakers, et al. ROOT- An Object Oriented Data Analysis Framework, http://root.cern.ch/, 2001.

[3] Carlos García Argos, on behalf of the ATLAS ITk Strip Collaboration. The atlas itk strip detector. status of r&d. *ATLAS Notes*, 2016.

[4] N. M. Ravindra, S. Abedrabbo, Wei Chen, F. M. Tong, A. K. Nanda, and A. C. Speranza. Temperature-dependent emissivity of silicon-related materials and structures. *IEEE Transactions on Semiconductor Manufacturing*, 11(1):30–39, Feb 1998. ISSN 0894-6507. doi: 10.1109/66.661282.

[5] Jan Vlcinsky. Answer to: How can I make a time delay in Python? https://stackoverflow.com/a/23665492, 2014. [Online; accessed 28-August-2017].

[6] Michael Vollmer and Klaus-Peter Möllmann. *Infrared Thermal Imaging - Fundamentals, Research and Applications*. John Wiley & Sons, New York, 2011. ISBN 978-3-527-64155-0.