Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Don Mithila Meshan Palliyaguruge

Image processing for investigation of effects in Lithium battery electrodes

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Don Mithila Meshan Palliyaguruge

Image processing for investigation of effects in Lithium battery electrodes

Bachelorthesisbased on the study regulations
for the Bachelor of Engineering degree programme
Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the Hamburg University of Aplied Sciences

Supervising examiner : Prof. Dr.Ing. Riemschneider
Second Examiner : Prof. Dr.Ing. Lutz Leutelt

Day of delivery 6. März 2015

Don Mithila Meshan Palliyaguruge

Title of the Bachelorthesis

Image processing for investigation of effects in Lithium battery electrodes

Keywords

Lithium iron phosphate battery, electrical and optical measurements, cathode, binarization, color Intensity, histogram, multithreshold

Abstract

Inside this report, the electrochemistry taking place at the interface of lithium iron phosphate battery cathodes of different composition is observed and captured by a microscope camera. Software solutions have been developed to investigate optical measurements of cathode chemical reactions using image processing techniques. Methodologies, for instance UNIX filters, have been used to speed up the process of analyzing mass data. The objective is to prove the correlation between electrical and optical measurements of test cells to determine the lithium battery's state of charge. This has been proven using visualizing effects such as video graphics and graphs.

Don Mithila Meshan Palliyaguruge

Titel der Arbeit

Bildverarbeitung für die Untersuchung von Effekten an den Elektroden von Lithiumbatterien

Stichworte

Lithiumeisenphosphat-Batterie, elektrische und optische Messungen, Kathode, Binarisierung, Farbmessung, Histogramm, Multithreshold

Kurzzusammenfassung

Diese Arbeit behandelt die Beobachtung der Elektrochemie, die an der Grenzfläche der Elektroden von Lithiumeisenphosphat-Batterien stattfindet, mithilfe von Mikroskopkameras. Softwarelösungen auf der Basis von Bildverarbeitung wurden für die Analyse der Aufnahmen entwickelt. Verfahren wie beispielsweise UNIX-Filter wurden angewendet, um die Verarbeitung großer Datenmengen zu ermöglichen. Die Zielstellung, den Zusammenhang zwischen optischen und elektrischen Messwerten zu belegen, wurde im Rahmen der Arbeit erreicht und mittels graphischer Darstellung, auch in Videoform, ausgewertet.

Contents

List of Tables

List of Figures

1. Introduction

1.1. Motivation

High energy density and long cycle life are some major qualities that have drawn attention towards Lithium-ion batteries in practical devices including Smart phones and laptop computers etc nowadays, and there are slowly being introducing to mass produced hybrid and electric vehicles. In electrical vehicles, batteries with many cells are used to supply the power train. The battery is controlled by a battery management system which needs measurement data from each individual cell. Common battery monitoring concepts use electrical measurements only and estimate the battery state of charge based on a calculated battery model. These models based on electrical parameters have three major disadvantages:

- the cumulation of measurement errors by the integration of current (drift)
- the need of a rest period to reach a chemical equilibrium after load or changing
- long term changes of cell parameters due to aging.

For that reason the battery research group at *Hochschule für Angewandte Wissenschaften* (HAW) Hamburg is investigating direct methods to capture and observe the state of the cells. Members of the research group discovered an optical effect, which seems particularly promising for that purpose. For lithium iron phosphate cathodes the reflection intensity is observed and reproduced. The observations were done with microscope cameras and specialized test cells. From previous experiments a large amount of image data is available.

1.2. Introduction to Battery Research

During the recent past, strong attention has been drawn towards lithium ion battery industry with the invention of electrical vehicles. Major factors that concern in the industry is safety of batteries, durability and reliability. To achieve these goals an intelligent battery management systems are required. BATSEN research group in University For Applied Sciences (HAW) Hamburg has been formed to address this goal. This project is funded by German Federal

Ministry of Education and research (BMBF). BATSEN's focus is on development of wireless battery management systems for vehicles.

Positioning individual wireless cell sensors inside the batteries has broadened the scope of battery cell measurements. Optimal thermal coupling is a possibility to improve accuracy of temperature measurements. More importantly, this allows direct monitoring to the chemical state of the battery.

Given this context BATSEN has been successful so far in developing optical sensors for monitoring liquid electrolyte density of lead-acid batteries. This estimates the State Of Charge (SoC) of the battery. [20].

However, the main goal of the research group currently is to establish a complete optical and electronic sensor system into battery cells using LEDs and optical fibres to capture and evaluate optical effects of the cell within the battery and wireless data exchange with a central battery management system.

1.3. Problem Analysis of Battery State Estimation

Several methods for estimating the state of charge (SoC) of a battery have been used. These methods are commonly limited to convenient parameters such as cell voltage, current and temperature. Direct measurement, Specific Gravity measurements, Voltage based estimation and current base estimation (Coulomb counting) are wide spread methods that are currently in use.

Unfortunately, these methods are suffering from certain drawbacks. In all practical batteries, discharging current is not constant, but drops non linearly as the battery get discharged. This can be measured by integrating current over the time. However, to measure the charge, battery needs to be discharge over the time. In most applications, information about the charge is required without discharging the battery. Therefore, direct measurement method is not convenient.

Specific gravity method measures changes in the weight of the active chemicals in the battery cell. As the battery discharges, electrolyte concentration get reduced. This reduction in specific gravity of the solution occurs in proportion to the SOC. Measurements were taken using suction type hydrometers which is not convenient in accuracy.

Coulomb counting integrates the current flowing from the battery to external circuit to calculate the amount of energy drawn from the battery. But this method suffers from drift caused by integration of measurement errors and thermal loses.

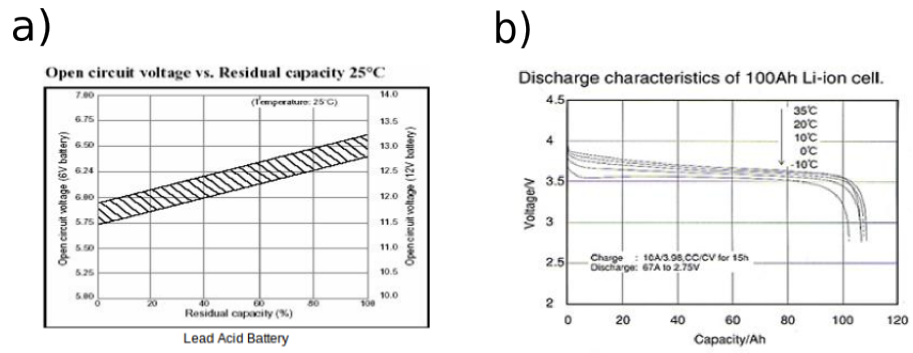


Figure 1.1.: *a)* Graph shows discharge rate for a high capacity lead acid cell at Open Circuit Voltage and the remaining capacity at constant temperature. *b)* Graph shows discharge curve for a high capacity lithium-ion cell.[2]

Figure 1.1 *a)* Shows the decline of open circuit voltage in proportion to the remaining capacity of the lead-acid battery. In contrast, lithium-ion exhibits a slow change in voltage during cell charge and discharge cycles (1.1 *b)*). Even though this shows ideal characteristic for battery applications, does not provide true measurements for SoC estimation.

2. Background and Literature

2.1. Lithium Iron Battery Technology and Chemistry

Battery in general, is a device consisting of many smaller batteries called cells. These electrochemical cells converts chemical energy into electrical energy. Each cell consists of positive and negative electrodes. They are called cathode and anode. These electrodes operate in a electrolyte chamber where ion can easily move between negative and positive terminals. This allows electron flow out of the battery to external circuits.

Batteries can be classified into primary and secondary forms. Primary batteries can be used only one time as the electrode material irreversibly changes during discharge. Secondary batteries can be charged and discharged multiple time as the original composition can be restored by the reverse current.

The following figure 2.1 shows the basic structure of a lithium-ion cell.

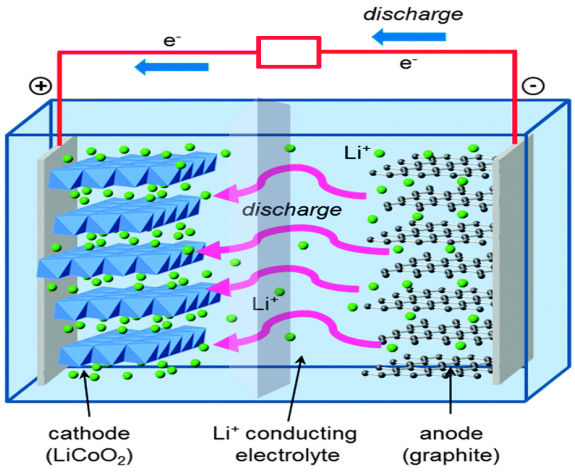


Figure 2.1.: Graphical representation of a rechargeable lithium ion battery cell. During charging $Li+$ ion flows to the negative electrode through electrolyte at the same time electrons flows out to the external circuit. In discharge this happens in opposite direction. Separator avoids short circuit between the circuit. Original image from [3] is modified.

As shown in the figure above, the electrical current reaches the cell via conductive surfaces, in most applications it's aluminium and copper either side the cell. The positive electrode (cathode) is made of pure lithium metal oxide. Uniform chemical composition of the material improves the battery life. The negative electrode (anode) is made of graphite, a form of carbon has a layered structure. Cell is filled with electrolyte which act as a transport medium for lithium ion to move freely between electrodes. For efficient charging and discharging can be achieve using pure and water free electrolyte.

The layer between electrodes called as separator which prevent electrical short circuit. It's made from micro porous polymer membrane which is permeable for small $Li+$ ions. During the charge, positively charge $Li+$ ions move from the cathode through the separator into the layered graphite structure of the anode. During the discharge, energy is removed from the cell. This causes lithium ion to travel back to the cathode. Then battery get discharged.

Positive electrode chemical equation [23].

$$Li_xFePO_4 \underset{Charge}{\overset{Discharge}{\rightleftharpoons}} Li_{x\text{-}1}FePO_4 + x \cdot Li^+ + 1 \cdot e^-$$

Negative electrode chemical equation.

$$C_6 + Li^+ + 1 \cdot e^- \underset{Charge}{\overset{Discharge}{\rightleftharpoons}} LiC_6$$

Overall chemical equation.

$$Li_xFePO_4 + C_6 \underset{Charge}{\overset{Discharge}{\rightleftharpoons}} LiC_6 + Li_{x\text{-}1}FePO_4$$

2.2. Optical Observation and Measurement Techniques for Battery State of Charge

As discussed in prior chapters, conventional methods that have been used so far in battery state estimation suffer from various drawbacks. Therefore, this has still become an open ended quest for battery researchers to find accurate solution for defining State of Charge (SoC) and State Of Health (SoH) of the battery.

Several academic researches prove uses of optical methods for analyzing ion transport effects and structural change. Even though this provide valuable information on the battery

state, practically this has to be done after disassembling the cell or in a controlled environments and expensive techniques. For instance glove boxes, electron microscopy or specially designed test cells.

Project BATSEN has developed a cost effective method for optical measurement reading during cell operation. Fibre optic sensor plat forms are used for observation of physical and chemical process within the battery cell. With the intervention of fibre optical sensor technique, research has divided in to three different directions.

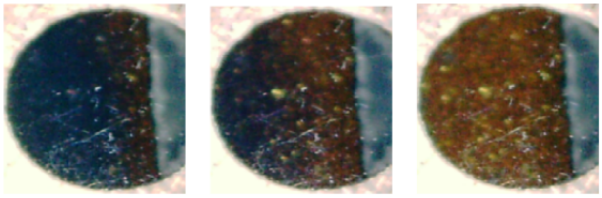
2.2.1. Optical Effects In the Electrolyte

During charging and discharging of a lithium battery, chemical changes are taking place over electrodes and electrolyte composition. For lead-acid batteries concentration of sulphuric acid is a direct indication of current state of the charge. Therefore, measuring density of the electrolyte has been tested using several techniques. However, most cost effective method can be recognized as Polymer Optical Fibre (POF) technique introduced by A. Cao-Paz [13]. It measures the acid density using transmission losses in the fibre during the battery operation.

In the Project BATSEN, a bachelor thesis has already successfully implemented this principal for lead-acid batteries. This continuously monitors optical characteristics of electrolyte during the battery operation using optical fibres and a sensor controller built in the cell. Due to the different cell chemistry, this measurement is not used in lithium-ion batteries.

2.2.2. Optical Anode Effects

Recently, a research group has discovered measurable optical changes on the graphite anode during charging process of the battery [18]. The intercalation of $Li+$ ions in the graphite turns gray black electrode into red-brown during the charging process. BATSEN has confirmed this scenario by an independent observation using test battery models.



a) Begining of the effect b) After 98 minutes c) After 373 minutes

Figure 2.2.: Photograph series describes the discoloration of the anode during the charging state of the Li-ion battery cell. Intercalation of $Li+$ ion into graphite results this situation. Fibre optics can be used to capture these effects for further analysis.

This effect takes place as a result of lithium ion intercalation into graphite electrode. Therefore, it is believed that this change in the graphite layered structure could be useful in fibre optical sensor measurements for battery life estimation.

2.2.3. Optical Cathode Effects

Similar observation that performed in anode has practiced on the positive electrode (cathode). During the tests *LiFePO4* has been used as the metal oxide for the cathode. The results were proved again a strong optical change on the cathode. The lithiation process that takes place in lithium ion phosphate electrode results in a chemical change from *Fe(III)PO4* to *LiFe(II)PO4*. This shift of concentration from *Fe(III)* to *Fe(II)* causes changes in ion bonds and crystal structure which leads to an optical absorption behavior.

The thesis topic **Image processing for investigation of effects in Lithium battery electrodes** has been introduced by BATSEN research team to prove the correlation between the cathode optical observation and electrical behavior using image processing techniques.

Results of the thesis will justify the measurements of this optical absorption behavior. Thin $(< 100\mu m)$ optical fiber can be implanted within the electrode to carry out the measurements. Optical fibre related techniques and theory are further described in the subsection 2.2.4.

2.2.4. Measurement Technology

Prior mentioned changes of crystal structures of electrodes absorb certain wavelength of the light. Wavelength specific effects can be measured using light emitting diodes. similar technology is already used in oxymeters in medical fields.

The device sends two wavelengths of light across the body part to a photo diode. Absorption of light at the wavelength is significantly different between blood loaded with oxygen

and blood lacking oxygen. Blood pressure can be measured taking the difference in the absorbency levels at each of wavelength [10].

This method is replicated into this project to monitor optical assonance at the electrodes. Implementation has been don using light conducting optical fibres and integrated System on Chip solutions (see figure 2.3).

Figure 2.3.: Wireless cellular sensors with microcontroller, which are suitable to be integrated in large lithium iron phosphate round cells. Results of the project BATSEN at HAW.

2.3. Digital Imaging

Digital imaging plays a large part in this thesis as it is all about image processing. Digital imaging basically means capturing light and transmuted into an electronic file. This has number of process working in between. Digital imaging can be classified by the type of electromagnetic radiation which attenuates when reflect off the object. These reflected waves carries information of the object. Then this information convert into digital signals by image sensors that process at the end by computers to produce the image as visual light image.

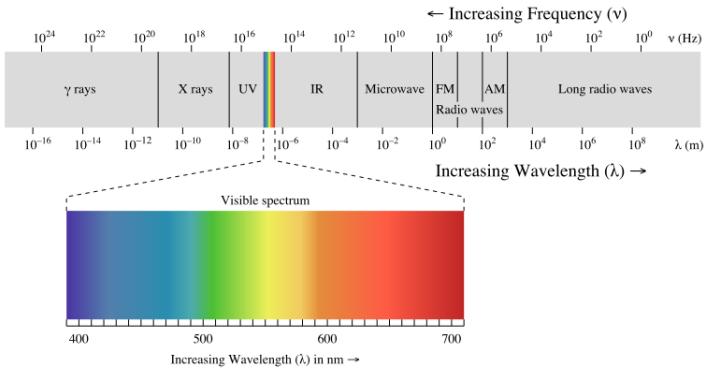


Figure 2.4.: Electromagnetic spectrum [8]

For instance, Gamma rays produce digital gamma ray imaging (digital scintigraphy), x-rays produce digital X-ray imaging, visible spectrum produce digital photography, sounds produce ultrasonography and sonar and radio produce radar [7]. Hence, Digital imaging brings broader prospect into image analysis using software.

Image processing for lithium batter electrodes effects has been carried out in the electromagnetic range of ultra violet (UV), Visible spectrum and Infrared (IR) in order to examine optical behavior in different mediums.

2.3.1. Microscopic Camera Settings

Digital microscope camera that have been used in the thesis, has a 9 mega pixel optical resolution with 10x to 200x magnification range. Low resolution images are possible without interpolation. Nevertheless, 2 mega pixel settings were used for all the optical measurements.

Digital microscope has its own inbuilt phosphor based white Light Emitting Diode (LED) light source. Emission spectrum of white LED is shown in graph 2.5. It can be seen that there is a significant proportion of blue luminescence which needs to be corrected. Since microscope has three color sensors (Red, Green, Blue filters) image will contain more blue channel information than the other channels.

Camera uses auto white balancing function or adjust exposure to rectify this issue. However, it is no known about the camera white balance function's behavior. Auto white balance setting can give different results in consecrative shots taken under identical conditions. This might generate unreliable optical data from lithium ion electrode optical effect observation. Therefore, it is decided to work with disabled white balance settings for optical measurement readings since the observation is identical in all measurement.

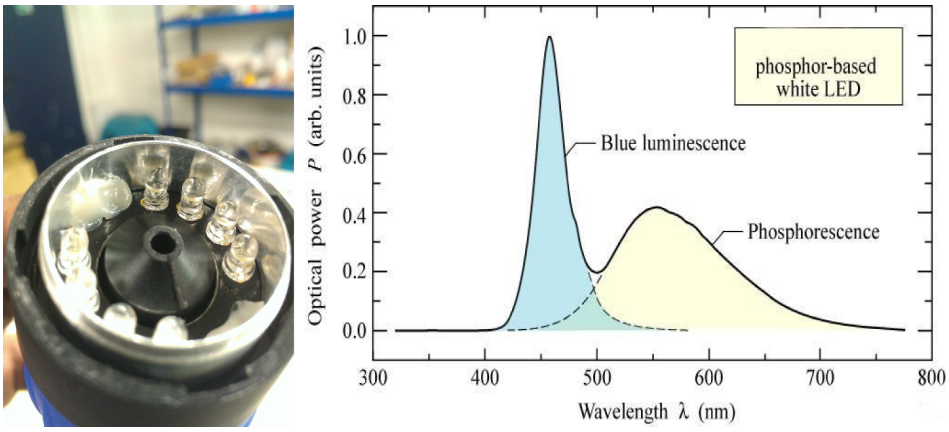


Figure 2.5.: Microscope camera lighting and Emission spectrum of white LED [11].

2.3.2. Image Sensors

Image sensor and filter information of the digital microscope is not known in the data sheet. The predecessor for this thesis topic, Mr.Greißbach has confirmed that this camera uses a Bayer mosaic color filters during his thesis. CMOS[1] and CCD[2] image sensors are mostly used in microscopy applications. Both chips sense the light through a similar mechanism called **photoelectric**. This occurs when light particles (photons) shines upon crystallized silicon to emit electrons. In a CMOS sensor, these electrons are collected and convert into voltage or transferred to metering register it's a CCD sensor. The measured voltage or charge is mapped to a digital electronic representation of the scene captured by the sensor using analog to digital converter.

A photo diode is a key element of a digital image sensor. It can also be refer as a pixel of the image. The microscope camera used in the experiments has a range of 3488 x 2616 pixels (9 Mega pixel). These pixels (photo diodes) are arranged in an orthogonal grid. The light gathered by the objective is focused by the projection lens on to the grid of photo diodes. These photo diodes are blanketed by ordered thin layer of red, green, blue dyed polymeric filters according to a mosaic pattern over the pixel array. The color is detected by passing the incident light across this polymeric filter. Figure 2.6 b) and c) shows this process.

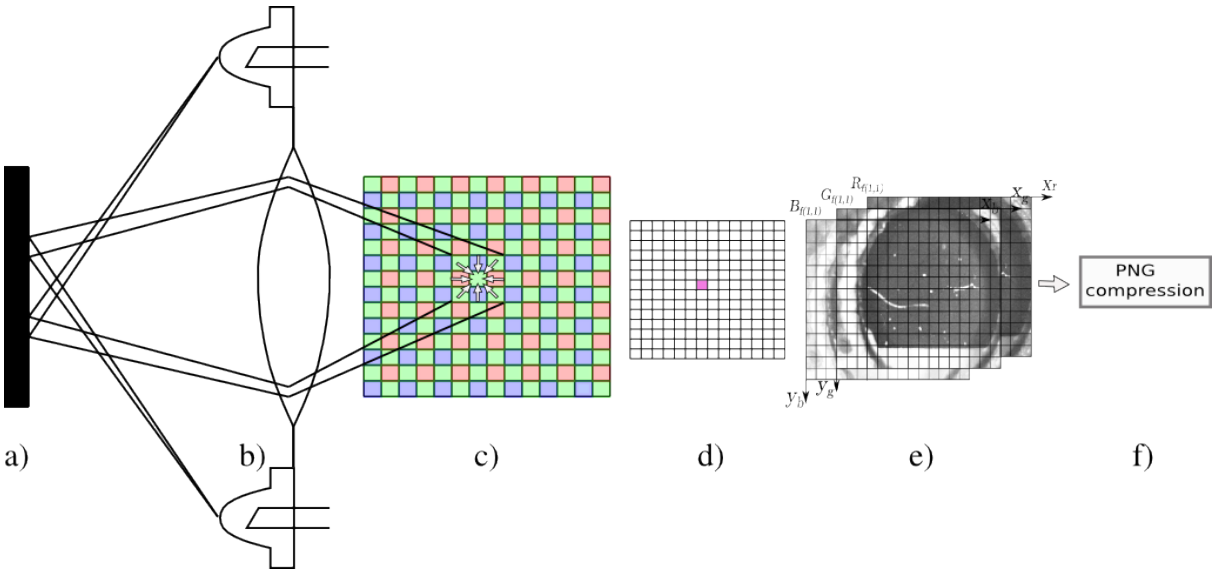


Figure 2.6.: a) Target Object b) Microscope optical lenses c) Bayer filter d) Camera sensor e) RGB color matrix f) Output file format

[1] Complementray metal oxide semiconductor
[2] Charge coupled device

The above mentioned polymeric color filter array is called **Bayer filter** found by Kodak engineer Bryce E. Bayer [5]. These filters are arranged in a pattern 50 % green, 25 % red, 25 % blue. Figure 2.6 c) illustrate an alternative filter rows. Every blue pixel surrounded by four greens and four reds. Every red pixel surrounded by four greens and four blues while green pixel surrounded by two reds, two blues and four greens. The reason for giving high emphasis on green is due to human visual response. Green located in maximum sensitivity wavelength region (550 nm) in the visible spectrum 2.4.

To generate the full color image, bayer output image needs to interpolate. Interpolation has been done using mathematical algorithm called demosaicing. This interpolation mechanism is not available with the microscope that are being used for experiments.

As for final step, image sensor output mapped to three channels (RGB) where each pixel has a color depth of 8 bits (256 colors).

Image processing in the thesis uses PNG[3] image file format which support lossless data compression. PNG supports palette based images (with palettes of 24-bit RGB or 32-bit RGBA[4]). PNG image can be made up of one or more channels. The number of channels are depends on whether the image is grayscale or color. PNG supports following combinations of channels.

Color type	Name
0	grayscale
2	red, green and blue (RGB true color)
3	indexed, indices for palette of colors
4	grayscale and alpha
6	red, green, blue and alpha

Table 2.1.: PNG color types [9].

[3] Portable Network Graphics
[4] Red Green Blue Alpha

3. Measurement setup

The following chapter describes the preparation process of the electrical and optical measurement setup for electrode observation on lithium ion battery while its operating. All the measurements carried out in the experiment based on modelled lithium ion battery environment using *ECC-Opto-Std* test cells.

3.1. Production of Test Cells

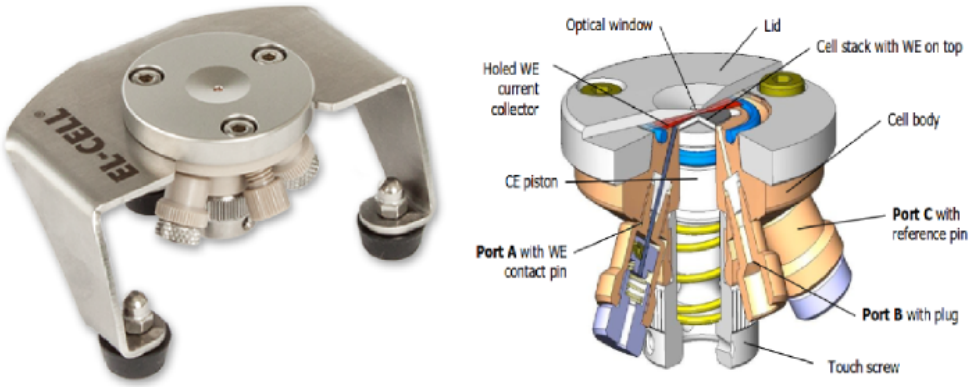


Figure 3.1.: ECC-Opto-Std Test cell [15]

ECC-Optp-Std is a model cell to conventional batteries in used. *ECC-Opto-Std* test cell is a production of *EL-Cell Gmbh* which allows monitoring the optical property changes of electrode materials during operating mode of the battery. The subject working electrode (WE), which is the cathode in this thesis experiments is placed right below an optical window. This allows to an optical instrument to capture electrochemical changes from the outside. The maximum diameter of the electrode is 10 mm and the inspection hole is about 1 mm of diameter.

As shown in figure 3.3 layers of the cell stacked as follows,

- Anode at the bottom (Lithium metal)
- Separator (Glass fiber)
- Rubber O-rings (Air tight seal)
- Cathode (Lithium metal oxide)
- Current Collector (Aluminum, Modified by cutting open the center for better visualization to cathode)
- Glass window (Top)

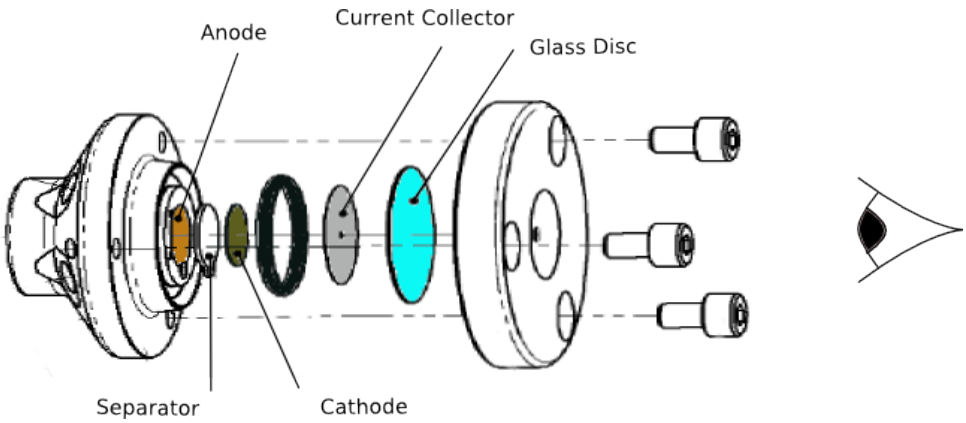


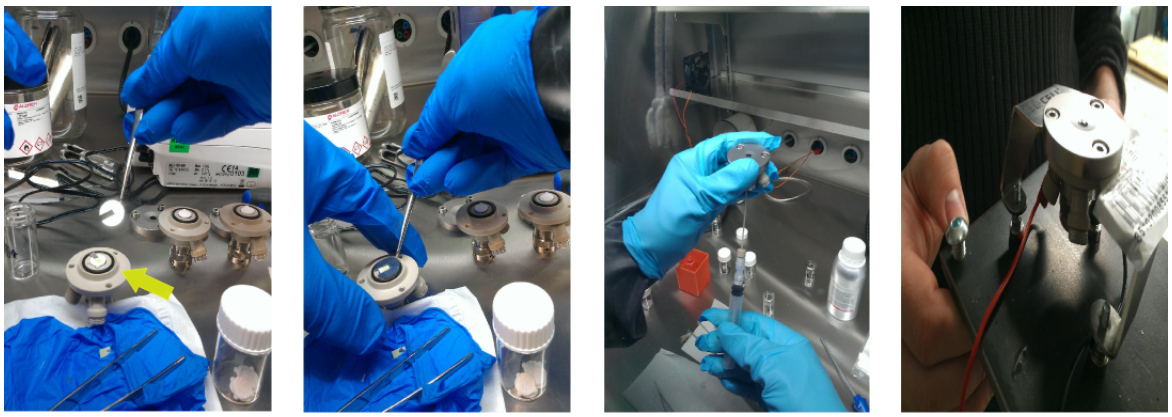
Figure 3.2.: Sandwiched structure for test cell parts

3.1.1. Cell Production Process

Cell assembly and filling process are carried out inside a glove box[1]. when cells are sealed measurements can be carried out in an ambient atmosphere. For easier optical analysis *ITO*[2] mixture with *lithium ion phosphate* has been used as cathode material.

[1] Glove box is a sealed container that is designed to allow one to manipulate objects where a separate atmosphere is desired

[2] Indium Tin Oxide



a) b) c) d)

Figure 3.3.: a) Glass fiber (separator) integration b) Aluminum current collector integration c) Clean electrolyte filling via the vacuum (syringe) method d) Air tight sealed test cell ready for electrical and optical measurements.

3.2. Data Recording

3.2.1. Electrical Data Recording

In order to prove the above mentioned cathode optic effect is unique, a laboratory test bench has created. To measure electrical measurements in this test plat form, following circuit setup is being used (see figure 3.4). For voltage measurements reading Fluke 45 is used whereas "DMM4020" is used for measure electrical current flow through the cells. The amount of current which in the range of nano amperes to several micro amperes depends on the cell structure/material that has been used.

The amount of current in a healthy test cell identified as around 3.5 μA during the tests. The measurements read out of "**DMM4020**" will be generated to text files name ending with,
"_zelle_1_spannung.txt" for Voltage measurements
"_zelle_1_strom.txt" for Current measurements

Measurements text file outputs can be generated by executing a bash shell script **startMeasurements.sh** design by Mr.Grießbach [17].

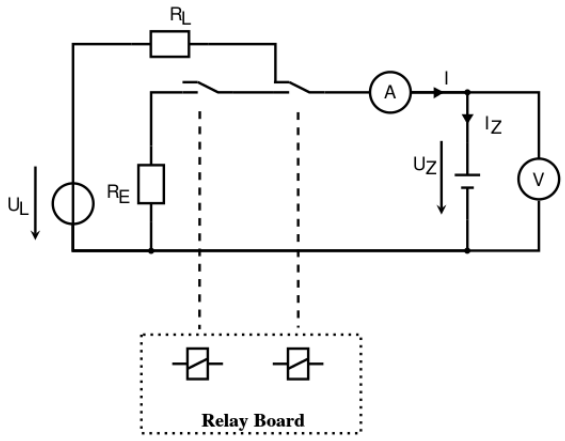


Figure 3.4.: Measurement setup for a one cell. Switches of the circuit are controlled by a relay board (PCB).

Electrical measurement preparing process

The raw electrical measurements that are being recoded over the weeks need to be processed before it takes for comparison with optical measurements. As measurement circuitry described in chapter 3.3.1 there's a systematic measurement error in the measurement in the cell current that we have to take into account.

So as the first step of preparing electrical measurements, current measurements need to be calibrated using the **CurrentCorrector** program (see chapter 3.3.1)

Step 1: Current Corrector Filter call
The Linux command:

```
# ./CurrentCorrector −u volts −l ohms −i ohms −c ohms < input current > corrected current
```

Second step would be to generate charge information of the cell. When electrical current flows through the battery for a certain period, we can calculate the respective charge data by performing mathematical integration of the current curve. This can be done by using **CurrentIntegrator** program designed by Mr.Grießbach.

Step 2: Current Integrator Filter call
The Linux command:

```
# ./CurrentIntegrator < input text file > output text file
```

Important European daylight Saving also a factor that had to be consider during measurements reading. As a solution for this we have taken UTC (Universal Time Coordinated) time as the time stamp for electrical measurements.

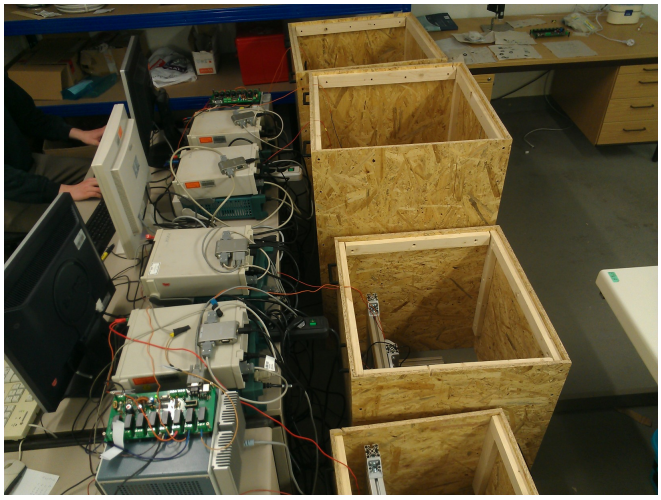


Figure 3.5.: Electrical measurement setup at the Li-ion Battery Experiment LAB. **Right** wooden boxes where microscopes and test cells are placed. **Left** computer units dedicated to record electrical and optical measurements. **Middle** Relay board which switches charge and discharge cycles in test cells.

3.2.2. Optical Data Recording

Image Copying Process

It is very important to chose a proper Operating system when it comes to handling large amount of data. In this project we have used Linux OS as it offers flexibility and possibility on fast data handling.
There are few important steps to follow in copying, sorting and arranging Image data on to the local drive. As described above, There are large amounts of Image data which are captured every 10 sec laps throughout.

Step 1 : File copying
The Linux Copy command:

```
1  # cp -p MR15Z4P2-*00.png ~/Work/MR13/MR13Z1P1/Images/
```

Basic function of CP command is to copy files and directories. It is mandatory to use $-p$ as the argument to preserve original timestamp of the image.

```
cp [OPTION]... SOURCE... DIRECTORY
-p same as -preserve=mode,ownership,timestamps
```

Step 2 : Pyrenamer - file renaming

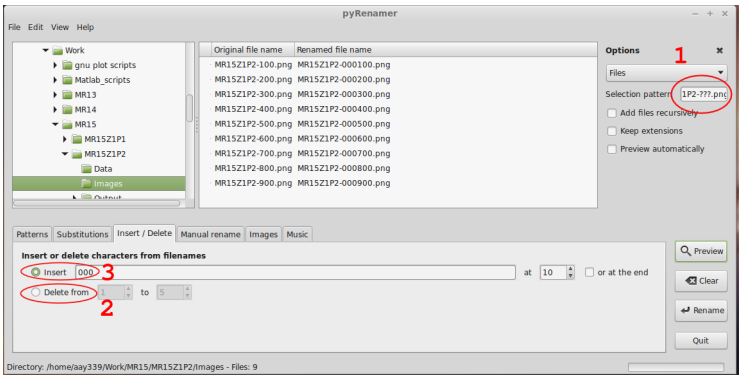


Figure 3.6.: Pyrenamer for File renaming: 1 - wild card characters to filter out images, 2 - zero removing from filename endings, 3 - zero padding to the front of the filename (ex: 00001.png)

Image files that were copied to the local drive need to be renamed. Zero paddings at the end of the file names has to be removed to perform video conversion in **Step 3** (ex: 0000100.png is not valid). This can be efficiently done using Pyrenamer program provided in LINUX OS system. This tool comes handy when handling Large amount of data.

Important Renaming needs to be done before move on to step 3.

Step 3 : Video making
The Linux video-make command:

```
1  # avconv -i MR13Z2P2-%04d.png -vcodec huffyuv -pix_fmt yuv422p output.avi
```

avconv is a very fast command line program for transcoding multimedia files. **avconv** reads from arbitrary number of input files which are specified by **-i** option and writes to an output file.

```
avconv [global options] [[infile options]['-i' infile]]
{[outfile options] outfile}
```

`-vcodec` set codec type
`-pix_fmt` Video color space specifier

3.3. Software Design

3.3.1. Current Measuring Tools

Current Corrector

Referring the electrical circuit setup for the test cell shown in figure 3.5, VDC measurement readings were proved correct and it can be utilized for cell electrical data analysis. Current measurements are subjected to a systematic error due to the circuitry. As shown in the figure 3.5 part of the measured current **I** flow through the voltmeter. Therefore, the amount of measured current **I** is greater than the true current(I_Z) that flows through the cell during the charging phase. Opposite scenario occurs in discharging phase.

To rectify this error a ready made UNIX filter design by Mr.Greißbach has been used for current measurements in this thesis.

The linux command call:

```
CurrentCorrector -l ohms -u volts -i ohms -c ohms < inputfile
> outputfile
```

Parameter	Description
-l	Discharge resistor in ohms
-u	Charging voltage in volts
-i	Internal resistance of the voltmeter in ohms
-c	Charging resistor in ohms

Table 3.1.: Current corrector parameters

The parameters used for the filter program "CurrentCorrector" are shown in Table F.1.

3.3.2. Timestamps Matcher

In video creation, timestamps plays a vital role in coupling electrical and optical measurements to form meaningful information on the video. Optical measurements in each video frame should be matched with its corresponding electrical value (voltage, current and charge) by timestamps.

Electrical measurements are recorded in every 1 second while optical measurements are captured in every 10 seconds. Therefore, a mechanism is needed to filter out electrical

measurements that are only matched with optical records. The implemented mechanism to do this task in this thesis called **TimestampMatcher**.

This tool has been designed using UNIX filter concept for easy execution for large amount of data. The respective *C* code can be find in the appendix C.8.

The linux command call:

```
./TimestampMatcher -f timestampfile < infile.txt > outfile.txt
```

3.3.3. Relay Board Loop Implementation

The usage of a relay board in this electrical setup is to establish a fully automatized resting, charging and discharging routine in a cell. Switching between these states can be controlled by commands that are transmitted by a computer via USB or RS232 serial interface.

Relay board specifications:
contains eight relays with changeover contact. Relay load 230 V/AC, 16A. The board is connected via a 9 pole null modem cable. Up to 255 relay boards can be cascaded in series.

Each relay card is activated by an address. First board receives its address via command "1" where each additional boards get addressed by increasing number of 1. So the complete command feds into the relay board consists of 4 bytes called a "frame".

Frame Structure

Byte	Protocol
Byte 0	Command
Byte 1	Board address
Byte 2	Data
Byte 3	Check sum (Byte 0 XOR Byte 1 XOR Byte 2)

Table 3.2.: Relay board Protocol. More technical data of the board, see the manual [14]

Charging and discharging of the test cells are being controlled by the input command file to the relay card. These commands are written in a text file format. The key commands in the files are "wait X", "SetPort X1 Y1". "Wait" command holds the execution for "X" seconds where "SetPort" sets the output register of the card with the address of X1 with the value Y1. Y1 value correspond to the channel address that are being used. Channel values and its functions are listed in table 3.3. It is possible to switch the activity of multiple cells by simply adding channel addresses. The comments are denoted by "#" in the file, key commands

are case-insensitive in the program. An example of a command text file can be found in attachment D.1.

Channel	Channel address	Function
1	0000 0001 (1)	Cell 1 Discharging
2	0000 0010 (2)	Cell 1 Charging
3	0000 0100 (4)	Cell 2 Discharging
4	0000 1000 (8)	Cell 2 Charging
5	0001 0000 (16)	Cell 3 Discharging
6	0010 0000 (32)	Cell 3 Charging
7	0100 0000 (64)	Cell 4 Discharging
8	1000 0000 (128)	Cell 4 Charging

Table 3.3.: Channels and dedicated functions

The relay board program and the command file initially were design by Mr.Grießbach. The relay board program and the command file has been further modified and new features are added to address some issues that came out during the cell charge and discharge setup during the thesis. The cycle routine plan (command file) along with relay board C program is modified with two extra keywords "Loop" and "End Loop". Assigning an integer number X after the "Loop" simply creates a loop of a X number of iterations which conclude with the "EndLoop" key word. This has mitigated the effort to write large amount of text commands manually to create loop of charge/Discharge cycles in the test cells. This also has reduced the risk of recursive errors which possibly could happen during copy and paste process in text commands into the cycle plan. Example of a command text file with loop can be found in attachment D.2.

The modified Relay board C program source can be found in appendix C.7. The key command "Loop" implementation has been designed using a linked list concept in C program. Execution steps of the loop described graphically in figure 3.7 along with an example.

Program flow of the "relaisCardController.c" described in a Nassi-Shneiderman diagram below 3.8.

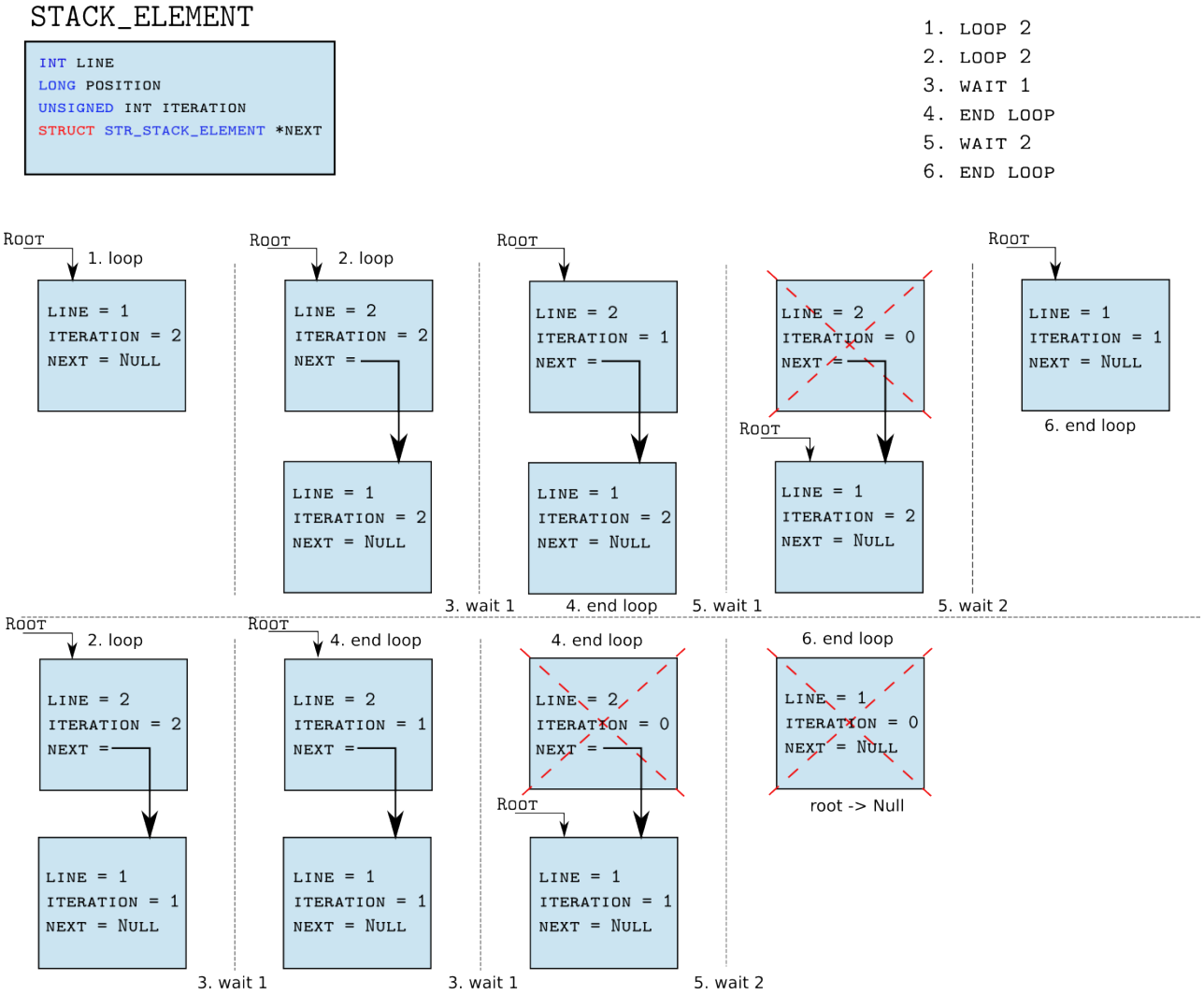


Figure 3.7.: Relay loop execution steps according to given example

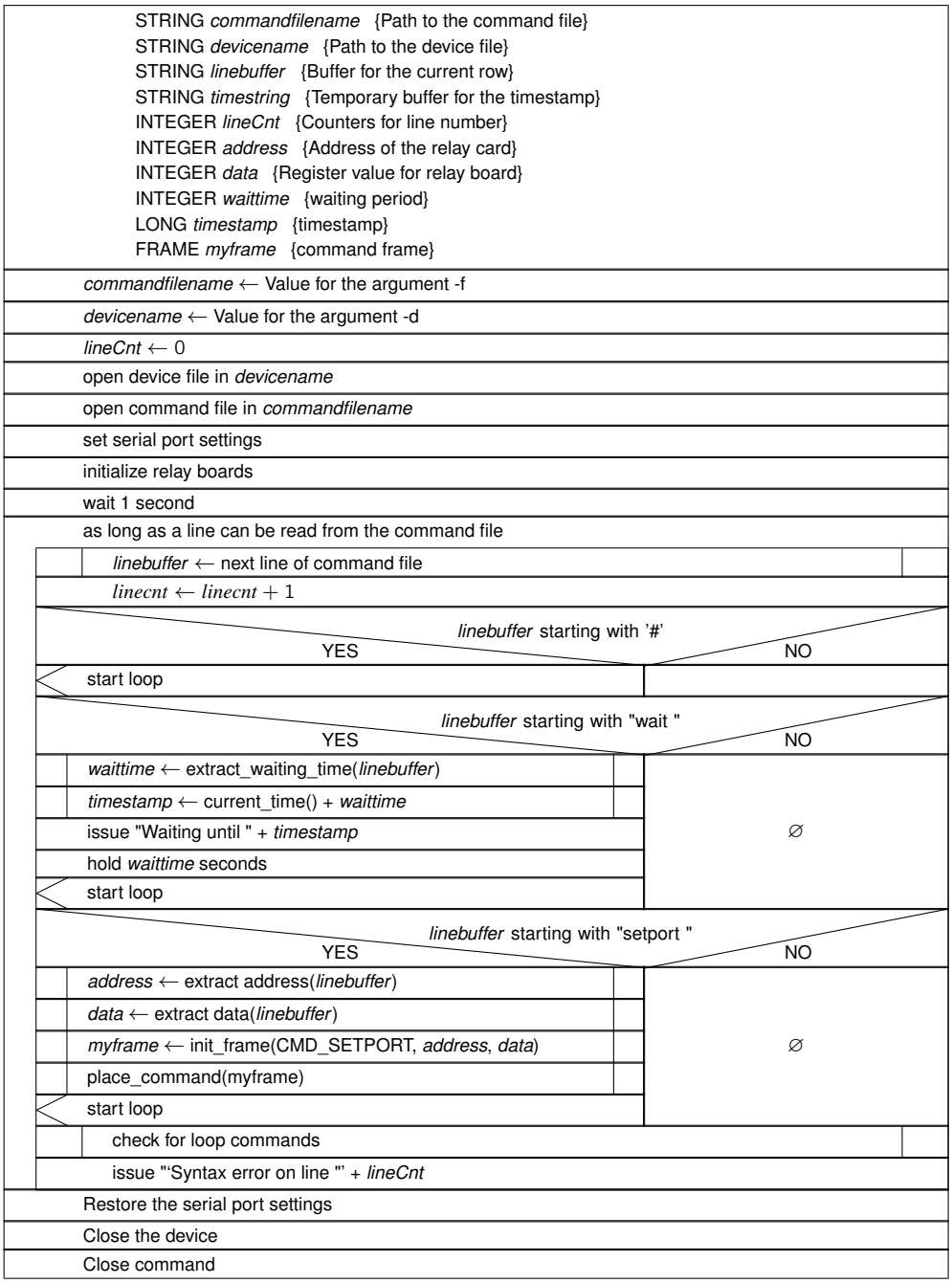


Figure 3.8.: Nassi Shneiderman diagram of Relay board controller [17]

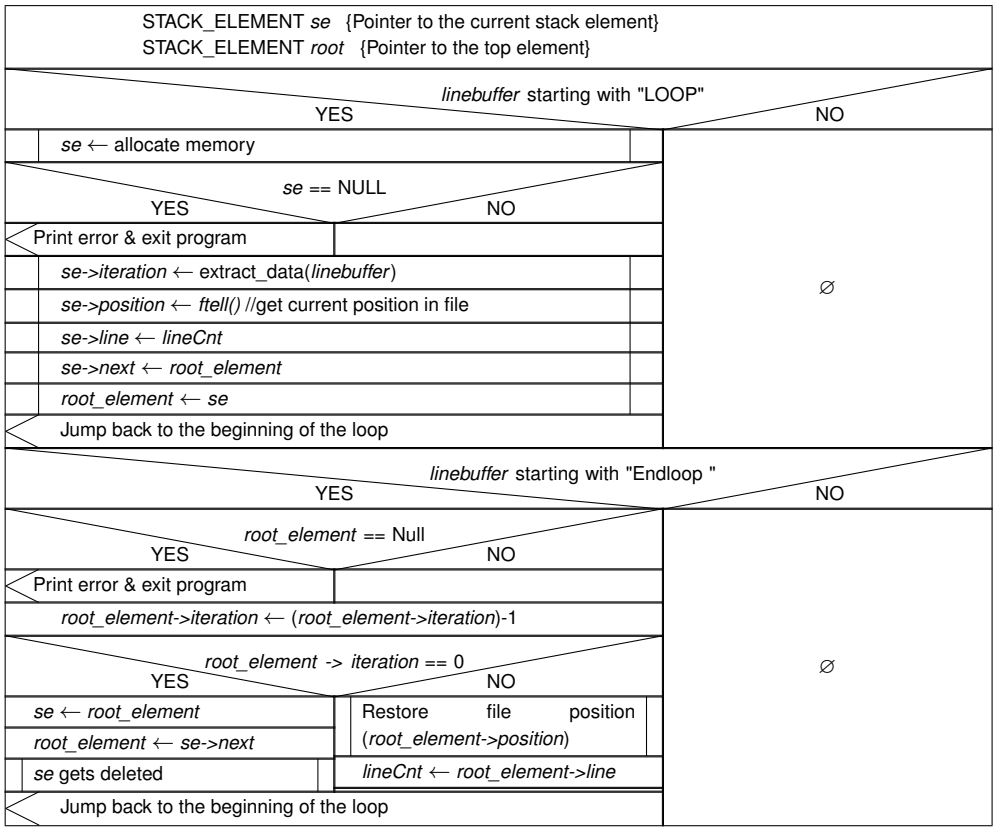


Figure 3.9.: Nassi Shneiderman diagram of Relay board controller part 2 (with the loop)

3.3.4. Battery Charging and Discharging Cycle Plotter

Setting up Cycle routine plan for the relay card is one of the major step in electrical measurement setup. The charging and discharging periods for each test cells are assigned through the cycle routine plan text file. In order to maintain a healthy battery cell life, charging and discharging periods need to be carefully decided. Over charging or undercharging would also bring instability to the electrical and optical setup.

Therefore, to support making decision on battery cycle periods, a visualization method for charge and discharge has been implemented. It's also a useful graphical interface for analysis individual cell charge and discharge patterns. An executable shell program is designed using filters in UNIX OS and command line graphical representing program such as Gnuplots to do this task.

This shell script designed using software pipelines in UNIX system (shell source code D.3). The Design of the filter is graphically displayed in figure 3.11.

This script reads in the cycle plan text file as standard input (see example D.2) and writes to a standard output file "cycle2gnuplot_output.txt" D.4.

The Linux command call:

```
1 cat cycle.text | ./cycle2gnuplot.sh (option 1)
2 ./cycle2gnuplot.sh < cycle.text (option 2)
```

The output text file from the filter visualized using Gnuplot script as shown in 3.11. The Gnuplot source code is attached.

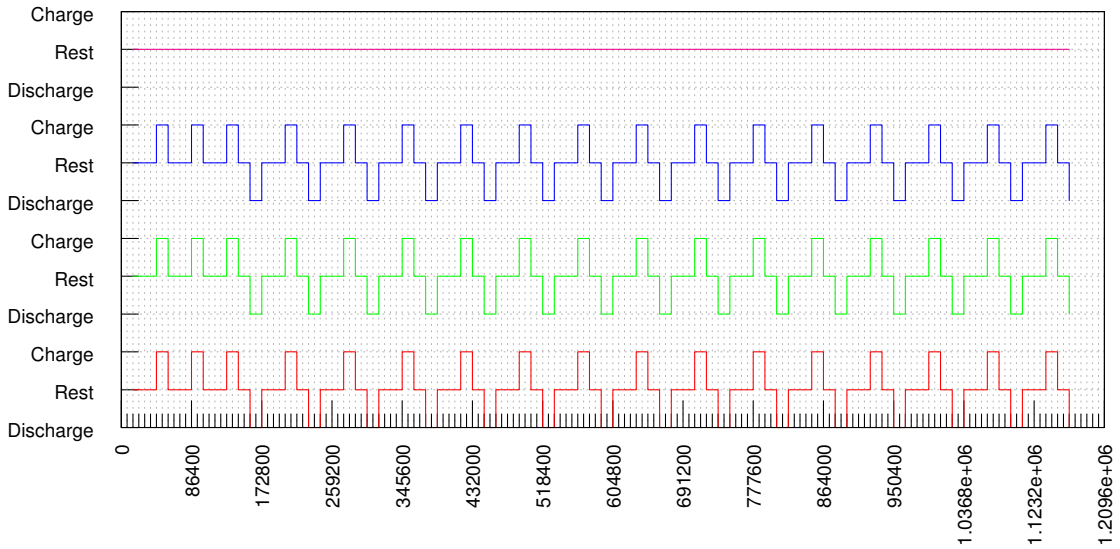


Figure 3.10.: Relay board cycle plan

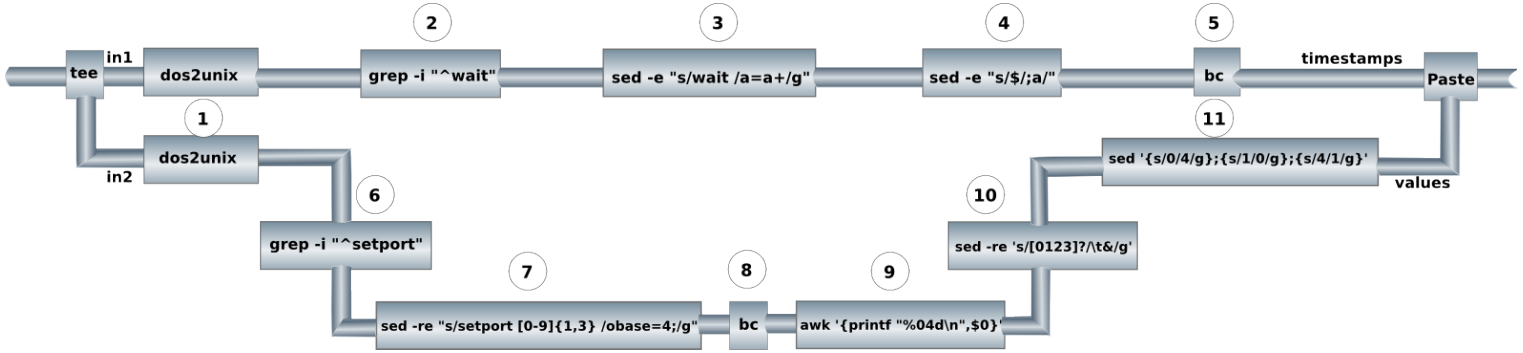


Figure 3.11.: UNIX filter design for Cycle routine plotter

1. convert line ending
2. filter lines with "wait", case insensitive
3. substituting "wait" with "a=a+"
4. substituting end of line with ";a"
5. arbitrary precision calculator, adds up timestamps
6. filter line with "setport", case insensitive
7. substitute "setport" and address with "obase=4;"
8. arbitrary precision calculator, translate input to base 4
9. zero padding (4 digits)
10. separation into 4 columns
11. switch 0’s with 1’s

3.3.5. Video Creation and System Commands in MATLAB

Video graphic creation is one of required tasks in this thesis. Optical measurements that were captured during certain amount of period will graphically be visualized together with corresponding electrical measurements.

Graphical videos were designed using **MATLAB** scripts and **avconv** video converters [1]. Video converter functionality is described in subsection 3.2.2.

Every single video frame has created using a MATLAB script C.3. Electrical data coupling into frames are done by executing UNIX commands within MATLAB by using **system()** MATLAB function. Key commands that were used described below.

The Linux commands:

```
1 [status dates] = system(['sed "1{/^#/d;};s/[\t].*//g" ' INPUT ]);
```

INPUT - Image timestamps source file (.txt format). This file is generated from MATLAB script "intensitaetsverlauf_kompensiert.m" C.2.

```
1 system(['cat ' INPUT1 '|grep -v "#"|./TimestampMatcher -f ./INPUT2.txt >' OUTPUT]);
```

INPUT1 - Electrical measurements (Voltage, Current, charge in .txt format).
INPUT2 - Image timestamps (dates - output from the first system command).
OUTPUT - Electrical measurements in matrix format.

Video creation steps are described in below table.

3.4. Flowchart of Complete Process

Step no.	Step description	Program and Parameters	Notes
1	...focus camera & picture	guvcview	
2	...take picture	guvcview	
3	...take electrical	startmeas (DMMreader.c)	
4	...copy	cp	
5	Correct Current	CC	
6	Integrate Current	CI	
7	"rgb-volt-current-charge-plotter"	TimestampMatcher	
8	video creation	avconv	

Table 3.4.: Chart of video making process

Step no.	Program and Parameters
1	guvcview
2	guvcview –no_display file=...
3	startmeas01.sh, startmeas45.sh
4	cp -p ...
5	CC -d \\ dev ...
6	CI
7	Matlab
8	Avconv

Table 3.5.: Table containing programs and parameters for the steps detailed in tab. 3.4

4. Image Analysis

4.1. Intensity Compensation and RGB-Channel Data Recorder

To support the main objectives of the thesis is to examine physical and chemical process within the battery cell during its charging and discharging states. To simulate this event for observation, test cells and electrical measurement setups are arranged in Lithium-ion battery observation laboratory. Microscope camera were been used during the battery observation process. (see figure 3.5). These camera's focused steadily on cathode of the test cell 24 hours a day capturing every 10 seconds of possible electrochemical reactions. These visuals mapped into RGB color space via the camera sensor. Color images are nothing but measurements of intensity and chrominance of light that reflects from the object.

In the image processing context one of the basic step is to analyse this red, green, blue channel distribution in the image. The idea of utilizing this method on our measurements is to examine closely at regions on the cathode where we think electrochemistry take place and yield every useful detail which we can not capture from human eye.

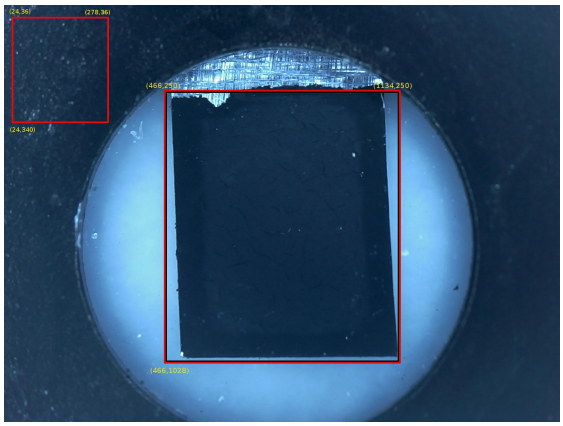


Figure 4.1.: Intensity compensation regions

During optical recording, it is assumed that the exposure of the camera has a slight difference from one image to another. Mr.Grießbach in the BATSEN battery research project introduced Intensity Compensation to rectify this issue.

Basic idea of this method is to select two regions, one from the area of interest and the other from an area which stay constant during the charging and discharging states on the battery cell, for instance, the depicted rectangular regions in the figure 4.1. As for the next step is to calculate average intensity level in each region to compensate later with one another. Software implementation of this procedure elaborated further in following section 4.1.2.

Auto white balancing control in the camera is another disturbance which can be addressed by this method. Auto white balancing shifts intensity level in different color channels. Therefore, compensation needs to be done for each color channel in selected regions.

4.1.1. Best Reference Region

The concept reference region selection comes along with Intensity compensation method. Reference region is basically selected from an area where electrode effect doesn't take place. Potentially useful areas for this purpose are follows.

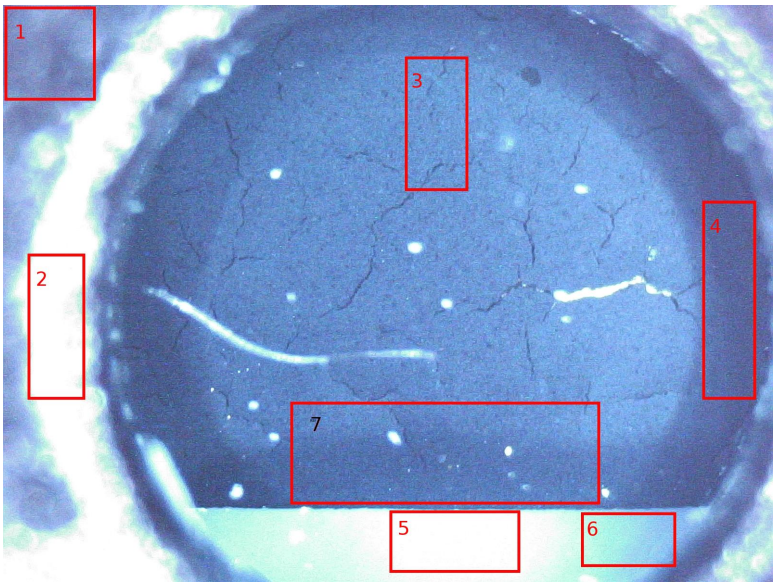


Figure 4.2.: Reference region selection for Measurement series 13 cell 3:
1) dark surface of the Cell lid 2) edge of the lid 3) cathode surface (excluding effected region) 4) shadowed region on cathode 5) Separator bright surface 6) Separator dark surface

RGB channel intensity from no.3 and no.4 regions displayed a slight correlation with the cathode chemical effect which suppress effected region (region no.7) intensity during compensation. 5th and 6th regions continued to bring influence of electrolyte in the compensation as its on the separator. Compensation proved much successful in stability of intensity with no.1 region compared to no.2. Therefore, dark region of the cell lid (no.1 region) used as the reference region for all the measurement series. The effectiveness of this technique in suppressing camera related brightness errors justifies in figure 4.6.

4.1.2. Software & Algorithm

Once the optical measurements placed in the relevant cell folder structure B.1, Intensity Compensation needs to be performed in these images(every 100th) of the measurement series. A MATLAB script is used to implement this process fully automatized which runs through all the optical measurements. Camera positioning on test cells window is different in each measurement series. Therefore, manual adjusting the coordinates of compensation regions in MATLAB scripts is not efficient task. This has been simplified by implementing general configuration file C.1 where coordinates, other required file paths can be set one time for each measurement series.

Basic intensity compensation mathod from MATALB script *"intensitaetsverlauf_kompensiert.m"* has been modified to adapt to this thesis. The algorithms used in the script are following,

Let captured Image series be *S*. Dimension of "Area of Interest" *M x N* and dimension of Reference area *P x Q*.

MATLAB interprets a image as function *A(x)*, *x* is a two dimensional vector represents image position. For a color image *A(x)* is a vector with three components RED, GREEN, BLUE. See figure 4.3.

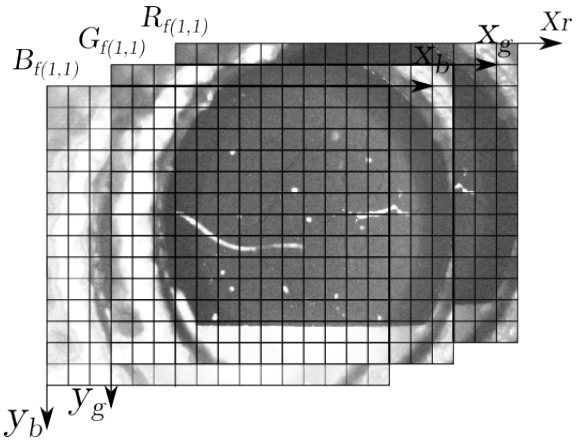


Figure 4.3.: Image RGB Matrix. Image is simplified for visualization

Assume area of interest begins at *A(xpos,ypos)* and mean value is $\mu_E[1xS]$ matrix

$$\mu_E(i) = \begin{pmatrix} \mu_{E,R}(i=1) \\ \mu_{E,G}(i=2) \\ \mu_{E,B}(i=3) \end{pmatrix} = \frac{1}{M*N} \sum_{x=xpos}^{M} \sum_{y=ypos}^{N} A[y,x,i]$$

Assume reference area begins at *A(xpos_ref,ypos_ref)* and mean value is $\mu_R[1xS]$ matrix

$$\mu_R(i) = \begin{pmatrix} \mu_{R,R}(i=1) \\ \mu_{R,G}(i=2) \\ \mu_{R,B}(i=3) \end{pmatrix} = \frac{1}{P*Q} \sum_{x=xpos\_ref}^{P} \sum_{y=ypos\_ref}^{Q} A[y,x,i]$$

Intensity Compensated RED, GREEN, BLUE channels stored in $C_R[1xS]$, $C_G[1xS]$, $C_B[1xS]$

$$C_R(i') = \frac{\mu_{E,R}}{\mu_{R,R}} . \sum_{j=1}^{S} \mu_{R,R}(j)$$

$$C_G(i') = \frac{\mu_{E,G}}{\mu_{R,G}} . \sum_{j=1}^{S} \mu_{R,G}(j)$$

$$C_B(i') = \frac{\mu_{E,B}}{\mu_{R,B}} . \sum_{j=1}^{S} \mu_{R,B}(j)$$

Following script generates two text files and a .mat file at the end of the execution.

"MRxxZxPx_RGBdata.txt" - contains $\mu_{E,R}$, $\mu_{E,G}$, $\mu_{E,B}$, C_R, C_G, C_B
" imageTimeStamps.txt " - contains timestamps of images

4.1.3. Results

The optical measurements have been collected for three weeks in MR15. The color intensity analysis for these measurements indicate that there is a sudden brightness change at the microscope on 10th of January. These kinds of optical disturbances distorts the true data. Color intensities of reference region and affected region analyzed separately in figure 4.4 and 4.5.

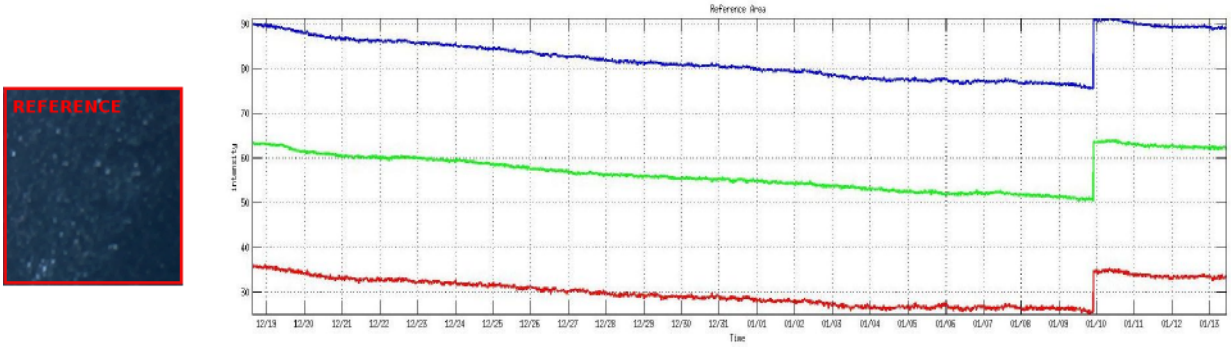


Figure 4.4.: Measurement series 15: Color intensity of Reference area at 25th cycle. Graph shows a sudden brightness change in effect on 10th January

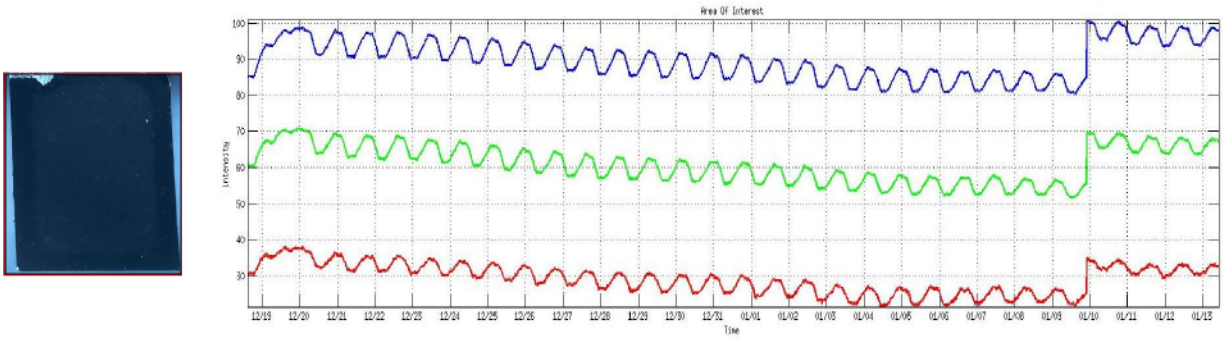


Figure 4.5.: Measurement series 15: Color intensity of affected area at 25th cycle. Graph shows a sudden brightness change in effect on 10th January

Figure 4.6 is the final output of intensity compensation method. It can be clearly seen that

the brightness error occurred in the preliminarily measurements are completely rectified and stabilized for rest of the measurements.

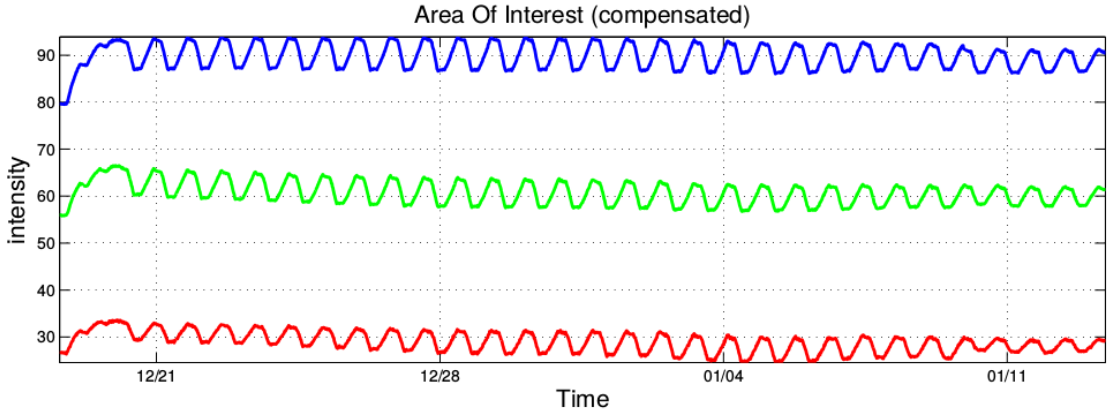


Figure 4.6.: Color intensity of affected area after compensation at 25th cycles. Brightness change in microscope has rectified.

4.2. Binarization

The image series below 4.7 is taken from different cycles in the measurement series can have different optical cathode effects over the time. These changes associate with the chemical reactions in the cathode material.

Figure 4.7.: **TOP:** MR15Z1 Cycle 5 **Bottom:** MR15Z1 cycle 25

Intensity Compensation method described in later chapter certainly gives useful information

about color space behavior of affected region on the cathode during charging and discharging states of the test cell. But these techniques alone itself doesn't provide any details of optical shapes that takes place on cathode, specially effects which can not be observed through naked eye.

Image binarization is a popular image processing technique which has numerous applications in astrophysicists, medical applications, printing industry. This method has been introduced to this thesis due to its image segmentation characteristics. The exact segmentation process that's been used is called "Otsu's method" [24] which assigns values foreground or background to pixel based on grayscale intensity. This allows visualizing a clear separation of active and non-active regions in the cathode surface.

4.2.1. Software & Algorithm

Basically binarization converts a gray scale image up to 256 gray levels into a black and white image. The simplest way to use image binarization is to select a threshold value and define pixels values which are above the threshold as 0 (Black) and others 1 (White). The challenge in this method is to find corresponding thresholds compatible to every image in the optical measurements series.

Syntax:

```
binaryim = im2bw(IGrayCropped_R, level)
```

im2bw function requires two arguments, grayscale image and threshold level. In this application, the affected cathode region has been chosen for grayscaled image(*IGrayCropped_R* - grayscale of RED channel). The output binary image *binaryim* replaces all pixels in gray scale image with luminance greater than defined "level" with 1 and all other pixels with 0. Threshold level specified in the range of 0 to 1. There is an inbuilt MATLAB function "graythresh" to generate this value using Otsu's method [24].

However, It has been decided to adjust threshold levels manually without "graythresh" function. As the focus is on entire charging/discharging cycles, threshold level should adapt equally and fairly for the entire optical measurement series. An unbalanced threshold level can lead to the loss of vital optical information on several batteries charging-discharging cycles in the series.

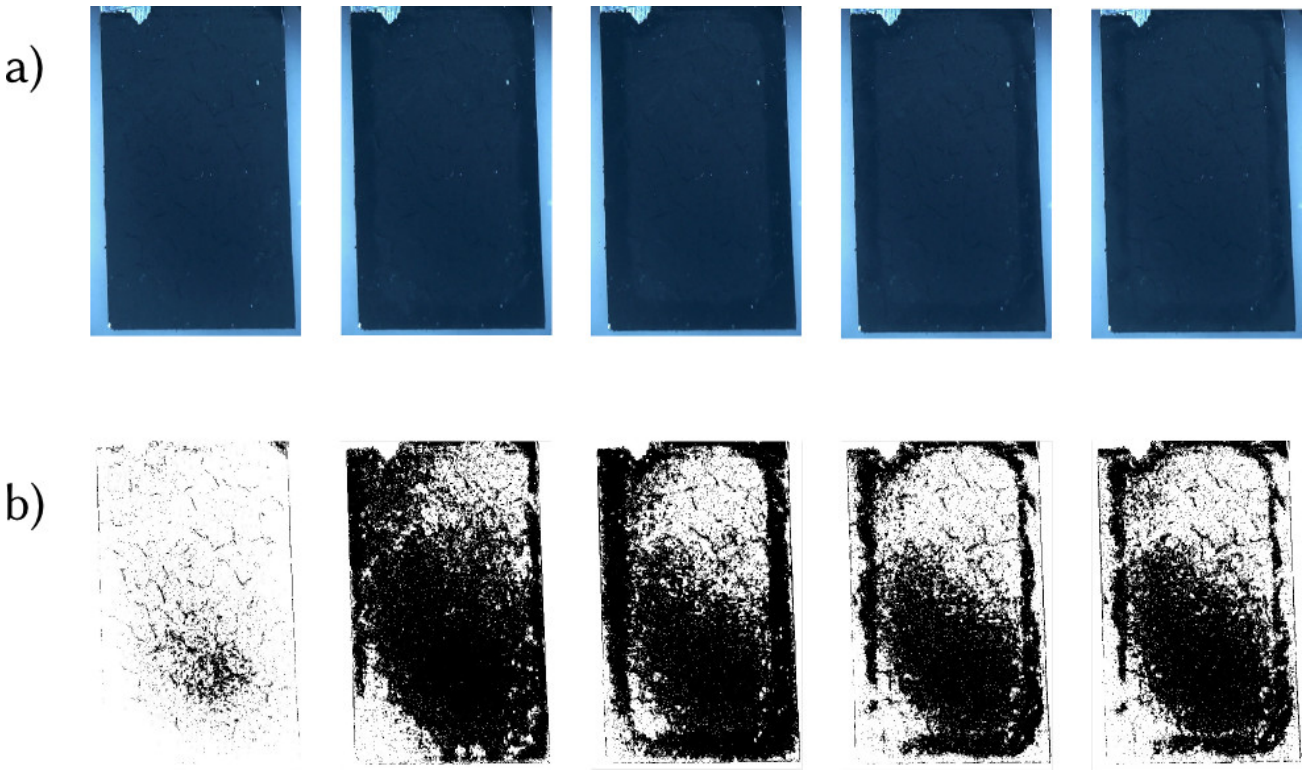


Figure 4.8.: MR15Z1P2 charge and discharge at 10th cycle. a) Original image with cathode chemical effect. b) Binarized version of (a).

Considering a fair threshold level, a MATLAB script C.4 has been designed to determine threshold levels focusing only on proportionally distribute battery cycles images in the entire image series that has clear active regions on cathode surface. Figure 4.8 (a) shows the highlight moments in 10th cycle of measurement series **MR15Z1P2**. 4.8 (b) shows binarized version of original images under a fairly chosen threshold level.

4.2.2. Noise Filtering

To address the main goal, Image binarized information needs to be recorded and compared with electrical measurements of the test cell and see for a possible relationship which can predict the battery state by optical means. Closer look at figure 4.8 (b) reveals extra information of cracks, shadows and possibly camera artefacts on the cathode. These unwanted information needs be masked out before binarized data be processed.

MATLAB Syntax:

```
B = medfilt2(binaryim,[span span])
```

medfilt2 is a non-linear function often used in "Salt and pepper" noise reduction. This function takes two arguments, input image as a 2-D matrix, and the neighbourhood matrix. neighbourhood matrix is window which spans along the image. In this implementation it's a square matrix size of *span*. In figure 4.9 explains further its algorithm.

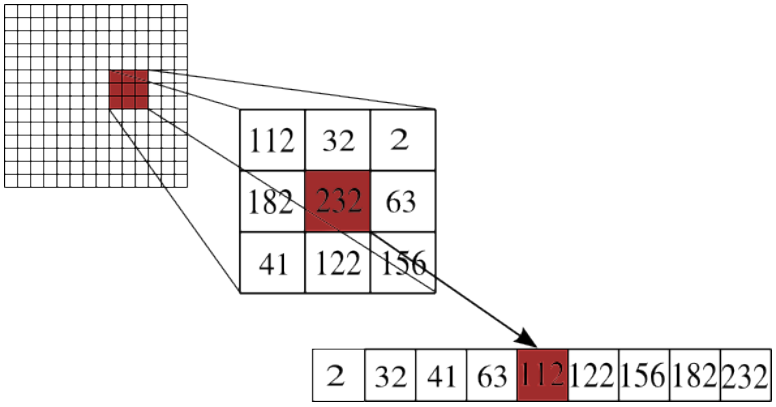


Figure 4.9.: In Median filter a window slides along the image and median intensity value in pixels within the window becomes the output intensity of the pixel being processed. [22]

Figure 4.10.: MR15Z1P2 Charging/Discharging 10th Cycle after Median filtering

So binarization is successful so far with median noise filtering. It is required now to transform this information quantitatively to be compared together with electrical measurements.

The following code snippet calculate the dark region as a percentage of the given area.

MATLAB Syntax:

```
% count dark pixels of a binary medfiltered image
numDarkPixels_R = sum(~B(:));
% calculating dark pixels percentage in the region
percentage_R(i) = numDarkPixels_R/totnumOfPixels * 100;
```

MATLAB script C.5 generates a text files and a .mat file at the end of the execution. **"Area_in_percent.txt"** - contains $timestamp$, $Channel\_R(\%)$, $Channel\_G(\%)$, $Channel\_B(\%)$

MATLAB script C.5 is a universal tool in the thesis to generate pixel count in dark regions(pixel value "0") of Optical Measurement series. For execution of this script **Config.m** and **imageTimeStamps.txt** files are required.

4.2.3. Results

The purpose of this method is to measure the growth of the affected section as a percentage out from the area of interest (large rectangle in red at figure 4.1). The result of this method proven a direct correlation with charge and discharge state of the battery (see figure 4.11).

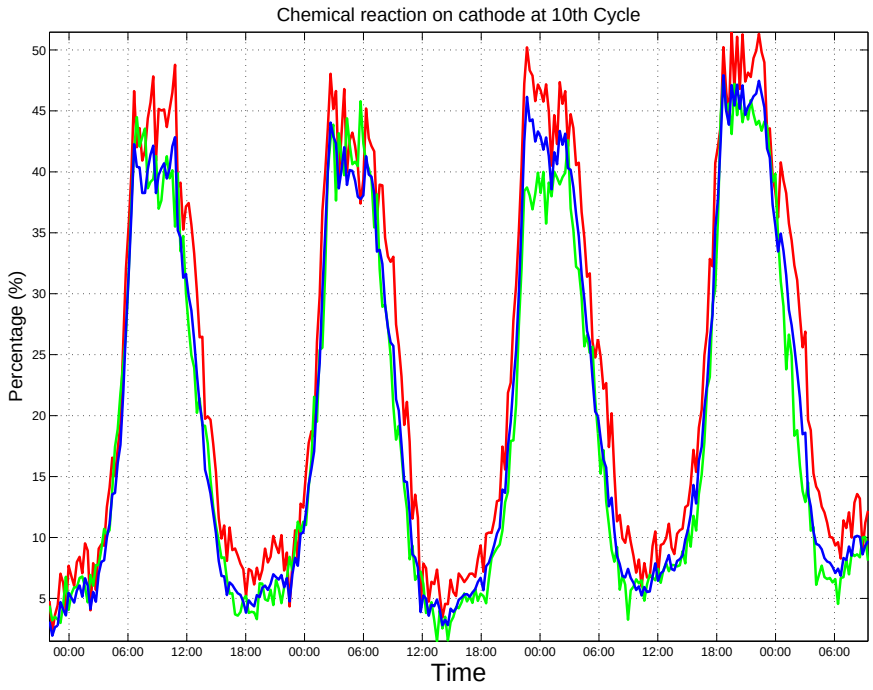


Figure 4.11.: Electro-chemical behavior on Cathode at 10th cycle in MR15Z1P2 measurement series. Data represent here as percentage of dark pixels that appear on binarized & noise filtered image 4.10. Red, Green, Blue lines represents dark pixel growth in RGB color channels

4.3. Multi-Threshold Algorithm

Binarization method is successful, so regarding visualization of active and non-active region on the cathode surface. As this method produce information only on two levels, questions arise about the information in intermediate levels in the image. It's possible to lose information due to single thresholding. Multithreshold technique solves above risen issues. This method utilize more than one threshold level to process the image. Therefore, more information can be extracted from images in comparison to binarization.

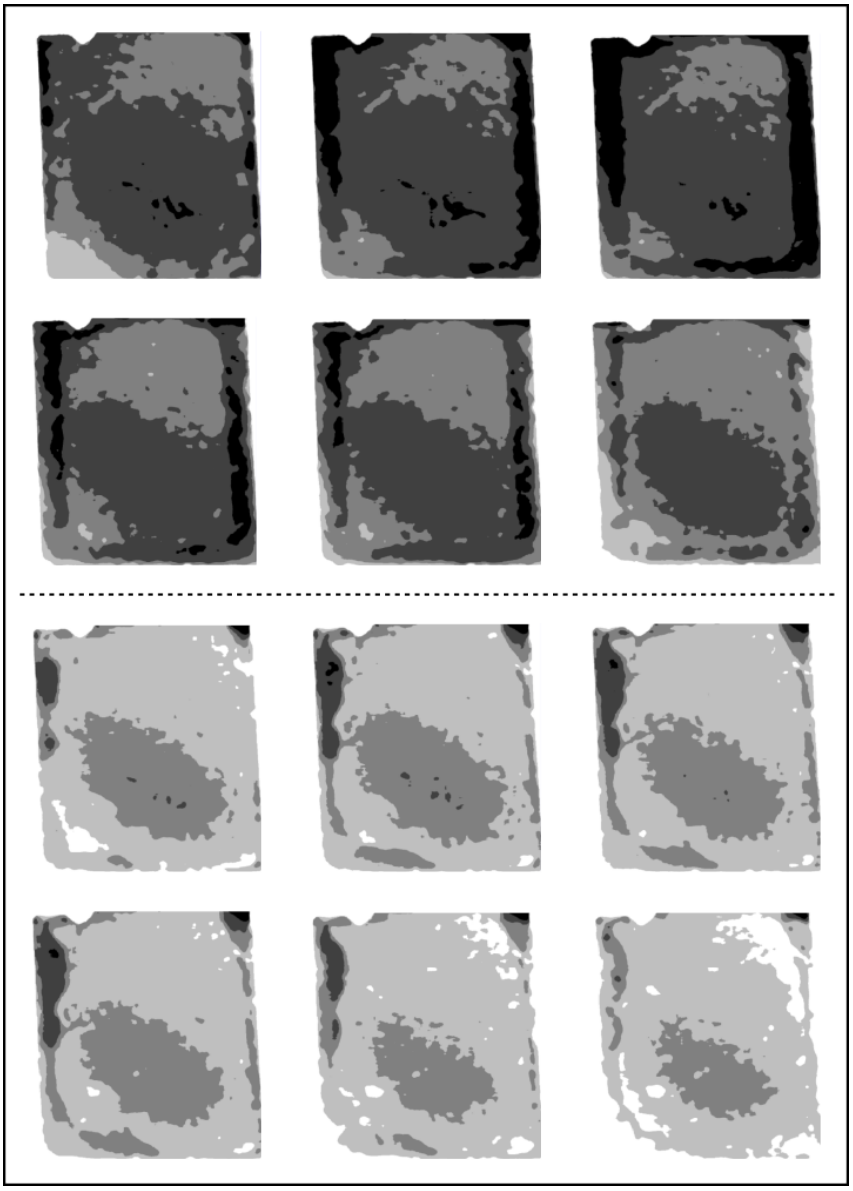


Figure 4.12.: **TOP:** MR15Z1 Cycle 1 **Bottom:** MR15Z1 cycle 25

The initial step of this method would be to define threshold levels which can be fairly applied for the complete optical measurement series. In binarization, these levels are picked manually by looking at beginning, middle and the end cycles of the image series. The low accuracy and inefficiency of this method leads to a systematic approach to find thresholds. Implementation of Multithreshold has been done using such a systematic approach for defining its threshold levels. Related literature for this method [25].

4.3.1. Software & Algorithm

The basic structure of Multithreshold method is implemented in two parts. As in Part 1: Finding threshold levels, Part 2: pixel count for each threshold level in the image. MATLAB script C.6 is designed for Multithreshold implementation.

Image Preparation: To eliminate noise and to focus in fixed image area following methods are used:
- *Imcrop ()* to crop out area of interest firmly from the image
- *Medfilt2 ()* to remove noise (Salt and Pepper)
The three RGB channels are always considered individually.

Part 1: Finding threshold levels

To determine the limit values of each image, the histogram of the three RGB channels of a fixed frame is analyzed. MATLAB function *imhist()* returns the histogram count *YR1_hist* and corresponding bin location in *a*. Since *YR* is a grayscale image, *imhist* uses bin value as 256.

MATLAB Syntax:

```
[YR1_hist a] = imhist(YR);
smooth(YR1_hist,spanSm,'moving')
```

smooth() smooths the data in the column vector *YR1_hist* using a moving average filter. Moving average denoted by 'moving' is a lowpass filter with filter coefficients equal to the reciprocal of the span. span is given 20 for this instance.

To eliminate disturbances, the averaged histogram values of three images are plotted. Further, a smoothing filter is applied to eliminate outlines of histogram plot. From the plot, the averaged and smoothed histogram peak values are determined. This process for determining the peaks of three averaged histograms is repeated for all the images. The peaks depend on the charge state intensity shifts in each of the three RGB channels. This shift is detected

by observing the peak. Based on the range of these peak values drift of the analyzed images, five equidistant gray values are taken.

$$threshs\_Red = [RminRmin + (Rmax - Rmin)/3Rmax - (Rmax - Rmin)/3Rmax]$$
$$threshs\_Green = [GminGmin + (Gmax - Gmin)/3Gmax - (Gmax - Gmin)/3Gmax]$$
$$threshs\_Blue = [BminBmin + (Bmax - Bmin)/3Bmax - (Bmax - Bmin)/3Bmax]$$

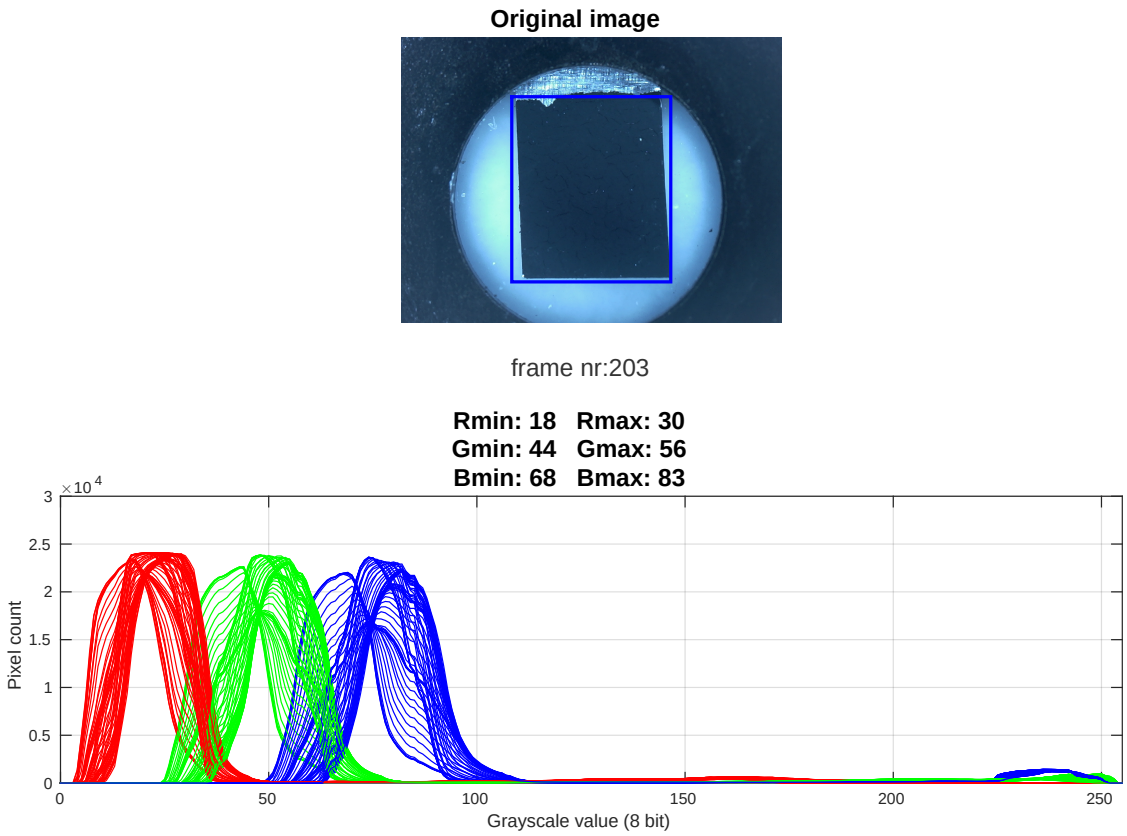


Figure 4.13.: Image shows, Histogram maximum peak drift in RGB color space in the effected cathode region (MR15Z1P2 charge and discharge 10th Cycle). This image is a single frame of 2229 frames.

Part 2: Pixel count for each threshold level

The five gray values picked from part 1 has been used in the second part to quantize the image into five sections. MATLAB function *imquantize ()* is used here.

MATLAB Syntax:

```
seg_IR = imquantize(YR, threshs_Red);
```

After quantization, the grayscale matrix is limited to numerical values 1,2,3,4 and 5. For better illustration of the images, these numerical values mapped to 8-bit common gray value range of 0 ... 255. In order to determine the area of each gray value range in the image, the pixels of different gray levels are counted.

MATALB script C.6 generates a text files and a .dat file at the end of the execution.
"multithreshOutput.txt" - contains $index, timestamp, X$ (X: total pixel count of each gray level (respective to the R, G and B channels))

"peakVals.dat" - contains $Rmin, Rmax, Gmin, Gmax, Bmin, Bmax$ (Peak values index in the range of 0..256 respective to Red,Green,Blue channels)

MATLAB script C.6 is a universal tool in the thesis to generate pixel count in each gray levels of Optical Measurement series. For execution of this script **Config.m** and **imageTimeStamps.txt** files are required.

Program flow of the respective MATLAB script is represented in following diagram B.2.

4.3.2. Results

This method has been introduced as a one step forward to the binarization method. Since the binarized image only provide two levels of data (Black or white), questions have arisen about the intermediate information on the affected region. It is possible to define number of levels that one would like to examine over the affected region by simply entering in MATLAB script C.6. In the example shown below consists of 5 levels of information at optical measurement Image number 2100 which belongs to 10th cycle of the battery state.

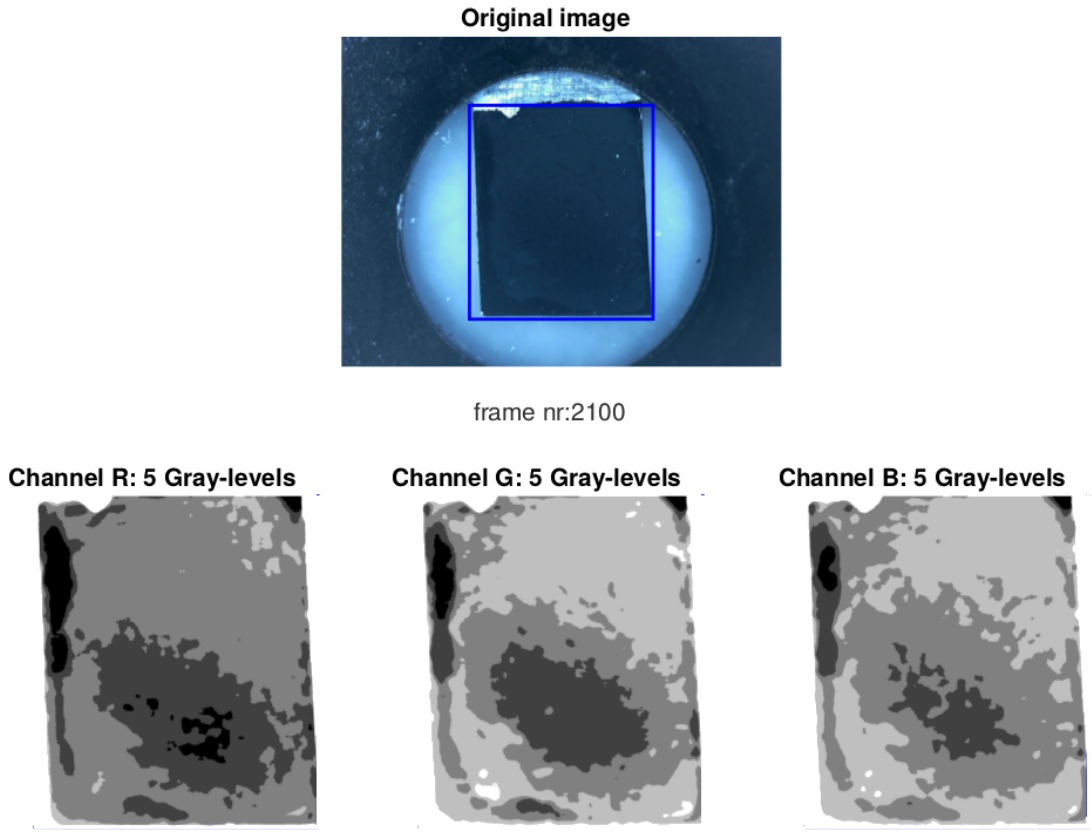


Figure 4.14.: Multithreshold of 5 gray levels (MR15Z1P2 Charging/Discharging 10th Cycle)

Pixel count for each levels (5 levels in this instance) for respective three color channels will be output to **"multithreshOutput.txt"** text file.

4.4. Edge Detection

Edge Detection is wide use technique and fundamental tool in image processing, Image pattern recognition, Image analysis and computer vision recognition. This technique detects edges in a digital image using mathematical methods at which the image brightness changes. Edge detection becomes quite useful in this thesis as its ability to verify boarder lines of chemically active regions on the cathode. Therefore, this technique draws the attention only towards the boarder lines regardless of other details of the image.

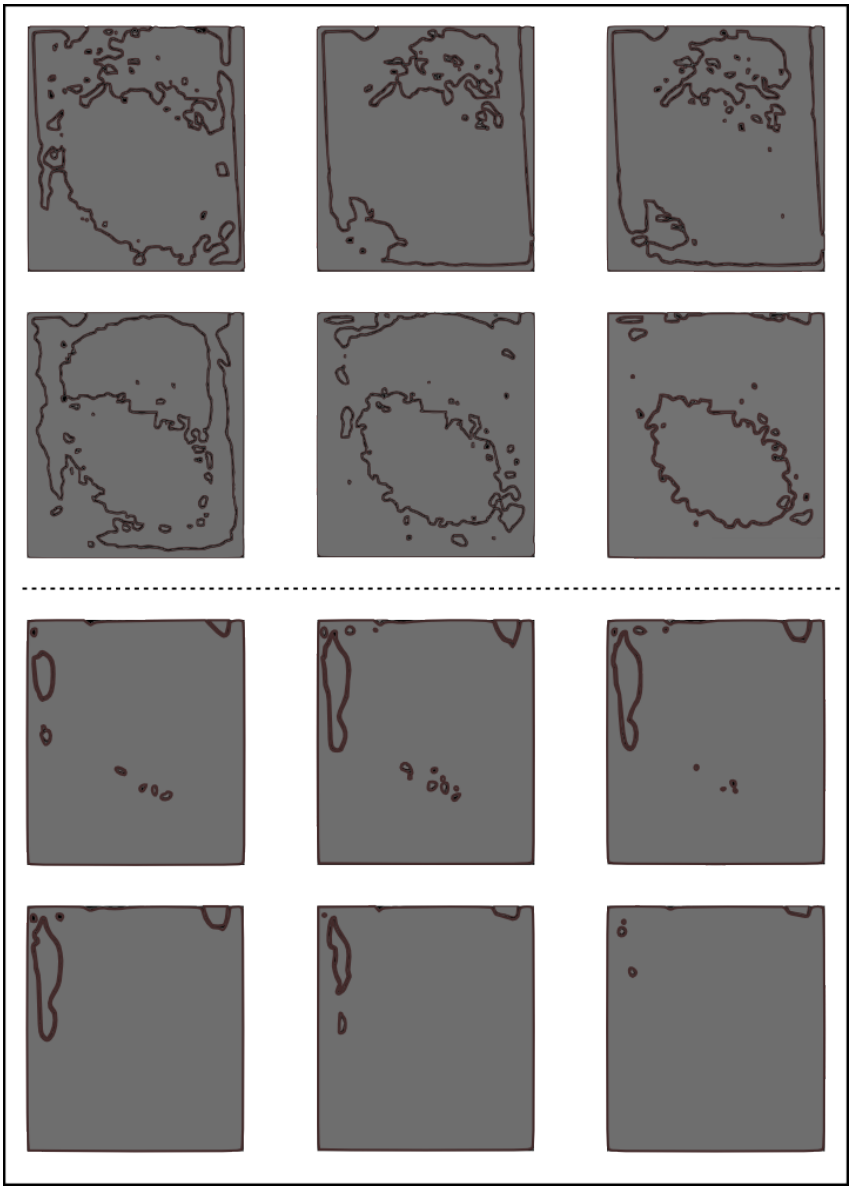


Figure 4.15.: **TOP:** MR15Z1 Cycle 5 **Bottom:** MR15Z1 cycle 25

In general, brightness can change due to discontinuity of depth and surface of the object, changes in material properties or illumination variations in the scene.

4.4.1. Software & Algorithm

There are several powerful edge detection methods available. "Canny" is known as the most powerful method out of all edge detection methods as it uses two different thresholds to detect strong and weak edges. The "edge" function in MATLAB is used here to perform edge detection tasks. The "edge" function searches for rapid changes in intensity in the image. This has been done comparing the first derivative of the intensity with thresholds or detecting zero crossings in the second derivative of the intensity [4].

MATLAB Syntax:

```
cannyim = edge(B,'Canny')
```

4.4.2. Results

This technique has been practiced as an experiment in finding boundary lines of the affected region on the cathode surface. The output image shown in figure 4.15 contains only thin edged boundaries (one pixel width) around the active regions. In the J.Harris experiments on motion of lithiation front in the graphite electrode based on the motion of color boundaries [18]. It is believed that these edge detection results may be helpful for successors of this thesis who are interested on performing J.Harris experiment on cathode side.

5. Mass Data Analysis

Following chapter is dedicated for evaluation and assessment of three sets of measurement series by its optical and electrical means. Only significant results and experiments in each measurement series are explained here. Full evaluation of electrical measurement data is shown in appendix F. All electrical measurements are corrected from systematic error described in 3.3.1. Abbreviations are used in naming of measurement series(MR), cells(Z) and subsections(P) for easy handling in mass data in this thesis. Image processing techniques are performed in images which obeyed specific microscope settings described in 2.3.1. A video is entitled for every measurement series to evaluate image processing results against electrical measurements.

5.1. Overview of Cathode Cell Structures

Several numbers of test cells were designed with different cathode material composition to improve optical effect on electrode surface. The chemical effect that takes place due to Li^+ ion intercalation in cathode of $LiFePO_4$ in a chain of chemical changes from $LiFe(II)PO_4$ to $LiFe(II)PO_4$.

Carbon (graphite,graphene) coating is a necessary factor for the improvement of electrical conductivity and the power density of a lithium-ion battery. It has turned out during initial tests that carbon is highly prohibitive in regard to optical analysis, as it strongly attenuates light. Therefore, a compromise has been made using indium tin oxide (ITO) which has both optical transparency and electrical conductivity. Table 5.1 and 5.2 shows several cathode material compositions that has been used for experiments in this thesis.

It can be clearly observed that healthiness of Electrical characteristics and sharpness of optical effects are clearly dependent on the composition difference of ITO and $LiFe(II)PO_4$. Table 6.1 summarizes this phenomenon.

MR	Date	Cell 1		Cell 2		Cell 3		Cell 4	
		Anode	Cathode	Anode	Cathode	Anode	Cathode	Anode	Cathode
13	27.09.2014	Li	V-2	Li	V-2	Li	V-2	Li	V-2
14	26.11.2014	Li	V-2	n.u	n.u	n.u	n.u	n.u	n.u
15	17.12.2014	Li	V-2	Li	V-2	Li	V-2	Li	V-2
16	20.01.2015	Li	VI-4	Li	VI-4	Li	LFP-Rep	Li	LFP-Rep
17	12.02.2015	Li	VII-2[1]	Li	VII-2[1]	Li	VII-3	Li	VII-3

Table 5.1.: Electrode material composition for each Measurement Series. (n.u. - not used)

Sample	NMP[2]/PVDF[3]	Carbon	ITO(mg)	$LiFePO_4(mg)$	$FePO_4(mg)$
V-2	629.2	n.u	198.5	494.9	n.u
V-3	626.8	n.u	397.5	500.3	n.u
V-4	626.9	n.u	199.1	n.u	500.6
VI-1	630.8	n.u	200.3	500.1	n.u
VII-2	615.9	n.u	194	490.4	n.u
VII-3	626.1	n.u	309.5	386.6	n.u

Table 5.2.: Cathode Material. (n.u. - not used)

[1] cathode is applied on ITO covered glass
[2] Solvent
[3] Plastic

5.2. Measurement Series - MR15

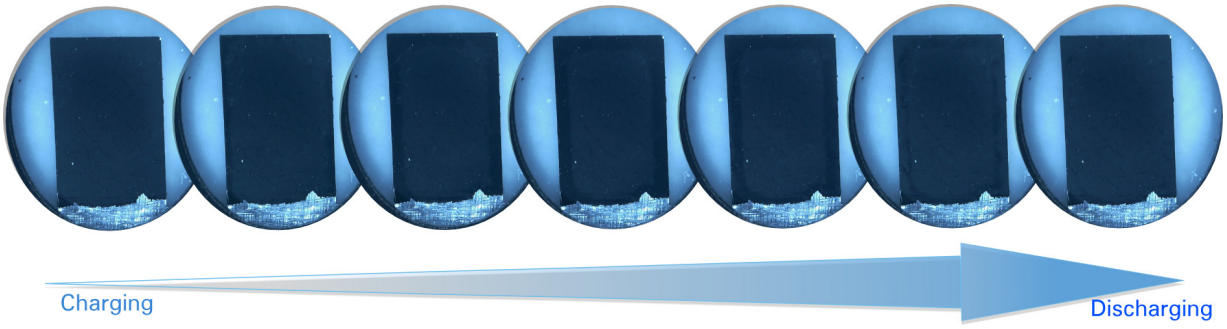


Figure 5.1.: Cathode effect: Measurement Series 15 Cell 1 at 10th Cycle

Cell no.	Total no. Of Images	Capture frequency	Processed Images	Frequency	Note
MR15Z1P2	222900	10 sec	2229	15 min per image	every 100th image
MR15Z2P2	222900	10 sec	2229	15 min per image	every 100th image
MR15Z3P2	223500	10 sec	2235	15 min per image	every 100th image
MR15Z4P2	223100	10 sec	2231	15 min per image	every 100th image

Measurement Series 15 contains data records for 4 cells that have identical electrode material structure (see table 5.1). All the measurement data for MR15 are dated approximately for one month of period. The basics of lithium ion batter technology is discussed in chapter 2.1. The positive electrode (cathode) is made as a mixture of *ITO* and $LiFePO_4$ (sample V-2). The negative electrode (anode) is made of lithium metal. During charing/Discharging process of the cell $Li+$ transfer through the transport medium, electrolyte (Lithium hexafluorophosphate solution). All the cells in the series follow the same charge, rest, discharge period pattern. Initially 4 hours of rest period, afterwards 3 steps of consecutive charging cycles lead the battery to reach its maximum open circuit voltage (F.9). Then Regular charge, rest and discharge pattern is followed until the end of the series.

Optical records were taken every 10 sec to capture Electro-Chemical behavior of cathode in individual cells. In this step, the whole electrode area can be seen through the glass window. This has largely contributed to stabilize the RGB Color intensity behavior compared to intensity behavior in MR13 where only a small part of the electrode material was observed (Intensity behavior of Cell 1 5.3). Also, this provides a broader prospect into the cathode chemical patters observation.

Cell 1 Optical measurements provides much stable optical patterns compared to cell 2, cell 3, cell 4 in the following series even though all the cells were equally treated.

This strong optical effect can be observed during intercalation of $Li+$ in the graphite as the blue cathode surface shadowed by dark-blue wave during the charging-discharging process and this is a recursive effect for all the cycles. Figure 5.1 shows this effect at 10th-12th cycles.

cell 2, cell 3 and cell 4 observations proves unexpected effect during its charging-discharging process. Thick-bright spots emerged in random pattern all over the cathode. This effect is not recursive during the rest of the cycles. Corresponding electrical measurements of these cells also displays drastic drop in voltage, current and charge curves which features a dying battery cell.

5.2.1. Intensity Behavior

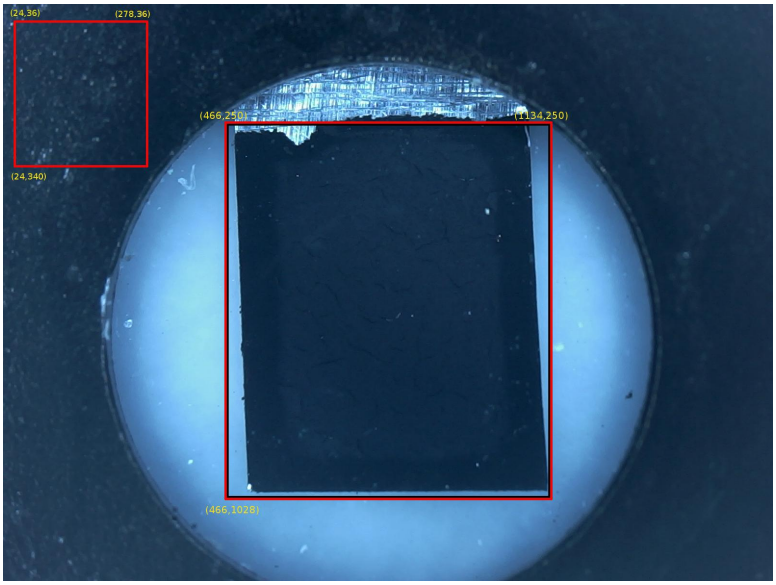


Figure 5.2.: Measurement Series 15 Cell 1 Part 2 - Affected region is highlighted by the large rectangular border while reference region high-lighted by the small rectangular border. Regions were coordinated for image processing purposes. Affected region's RGB intensity is compensated with the reference region's total RGB intensity to calibrate the camera brightness error.

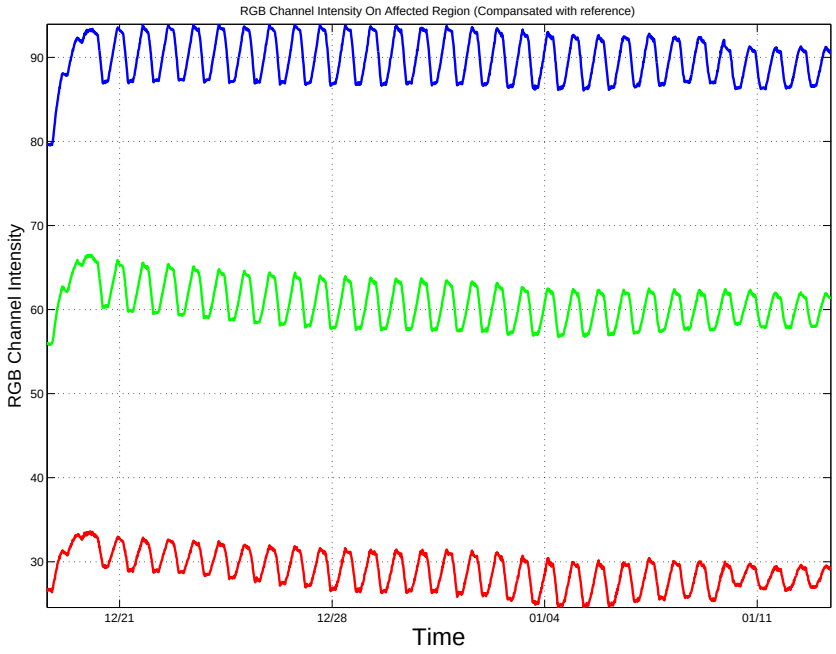


Figure 5.3.: Measurement Series 15 Cell 1 Part 2 - RGB Channel Intensity on the affected region (large red colored rectangular region in figure 5.2). These results are based on 2229 compensated images with the reference region

Measurement Series 15 cell 1 intensity data for Red, Green, Blue color channels shows much stable behavior compared to the other cells in the series. During the charging state cathode surface gets brighter whilst rest and discharging get dark blueish waves in the edges of the cathode. This correlation between charge and the color intensity further proved in figure 5.4.

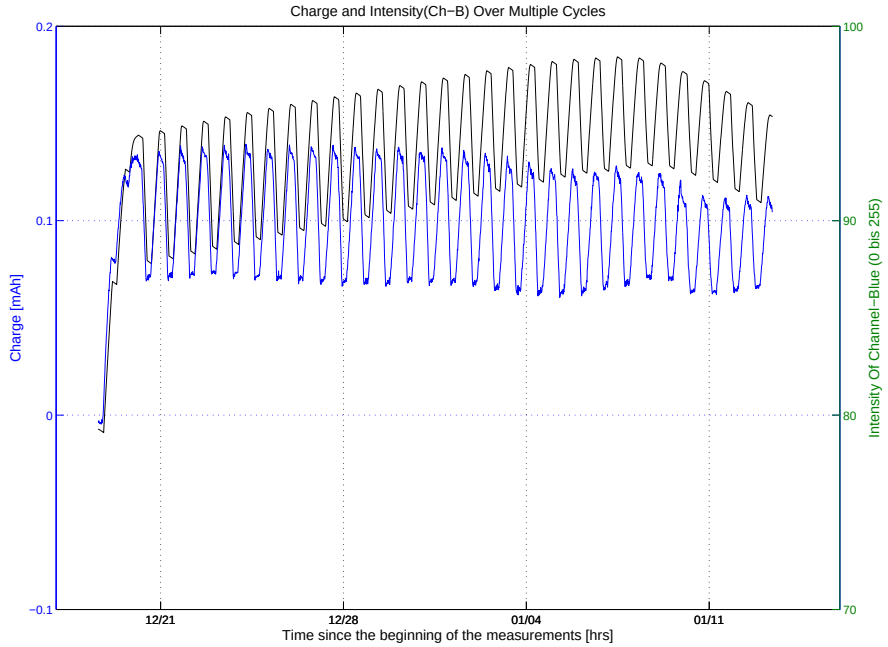


Figure 5.4.: Measurement Series 15 Cell 1 Part 2 - The correlation between cell charge and color intensity (blue channel in this instance) is described in the figure. Both curve proceedings are in phase which decline in a similar manner at the end due the decay of the battery.

5.2.2. Binarized Image Behavior

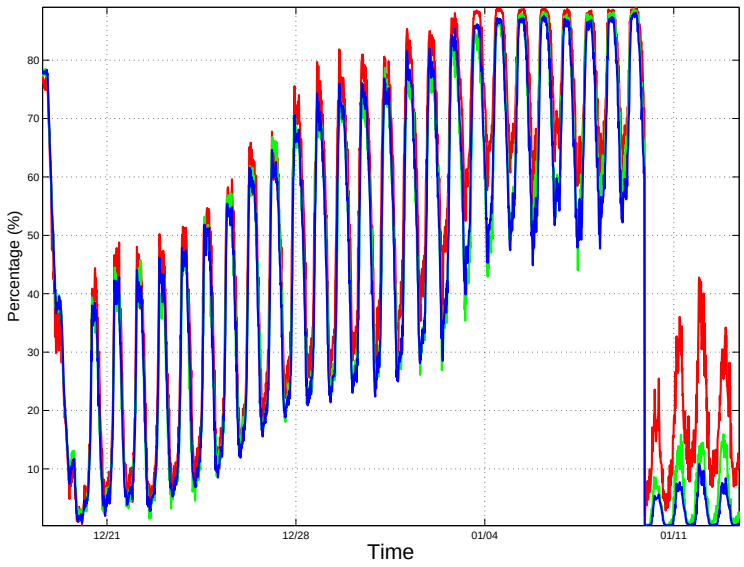


Figure 5.5.: Measurement Series 15 Cell 1 Part 2 - Binarization - Dark pixel growth on the affected region (large red colored rectangular region in figure 5.2) as a percentage. These measurements are based on binarized-median noise filtered 2229 images 4.10. Brightness change in microscope causes the sudden damp in the curve during the final phase of measurements.

Binarized image processing provides a different prospect of data in the affected region. Pixel count of binarized affected region (large red rectangular border in figure 5.2) considered to be 0 % during charging state of the cell where cathode has a clear blue surface. As described in chapter 4.2, growing dark-blue pixels in the affected region during resting and discharging states calculate as a percentage out of the whole region. Therefore, this shows a reversed polarity in behavior of charging and percentage graphs. This has clearly visualized in figure 5.6.

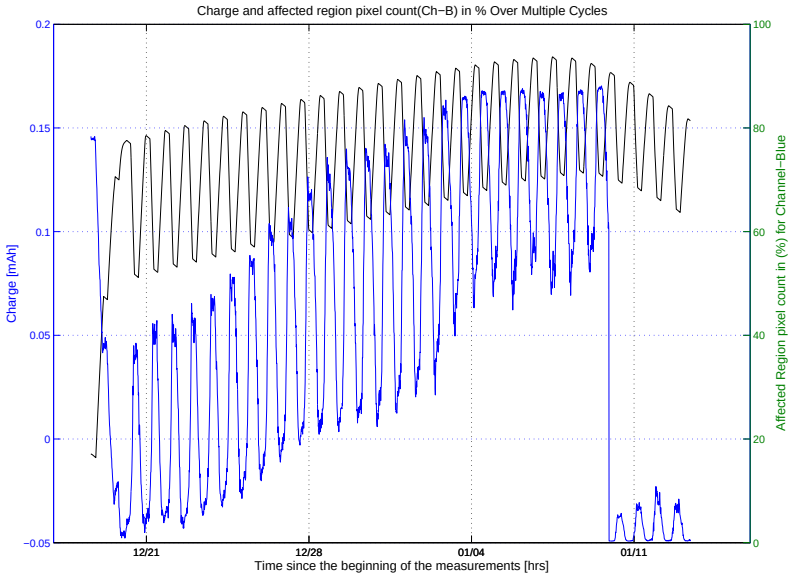


Figure 5.6.: Measurement Series 15 Cell 1 Part 2 - The correlation between cell charge and binarized pixel percentage (Blue channel in this instance) is described in the figure. Both curve proceedings are in reversed polarity. While charge graph shows a gradual decline at the end of the series, percentage graph depicts a drastic decline due to decay of the battery.

5.2.3. Measurement Series Interpretation

The final goal of this thesis is to find an accurate method to estimate the state of charge (SoC) of lithium-ion batteries. It would be easy if the battery could be charged or discharged at a constant rate. Unlike lead acid batteries, determining SoC is cryptic in lithium-ion cells. The image processing results discussed in this chapter describe the clear correlation between optical and electrical measurements in the test cells so far.

In order to estimate SoC of the battery, intensity results of MR15 cell 1 has been plotted against the charge. Figure 5.7 plot *a)* shows charge over uncompensated intensity which has a sharp drift away from its linearity. After intensity calibration with the reference region plot a) has been reached almost to its linearity (with drift) which is plotted in *b)*. This is a perfect example to prove the importance of the compensation method.

However, it has been discovered that plot b) measurements incur with two hours of time shift error (due to EU daylight saving). This has corrected in c). blue, Red, yellow and green colors in graphs represent every quartet of the measurements. Important fact reveals from

this four colors distribution is that, first quarter of measurements depicts much linearity and then a drift in mid quarters (blue and green). This drift becomes much stable in final quarters of the measurements which depicts by yellow and green thick linear lines.

The expected ideal plot would be a linear graph which all the intensity value has its corresponding charge. It is believed that this achieved linearity can be further improved by introducing more noise reduction techniques to image processing and using more sensitive and stable cameras. Result in c) graph proves that it is not far away from the ideal situation with more noise reduction techniques and more accurate techniques for measurements recordings.

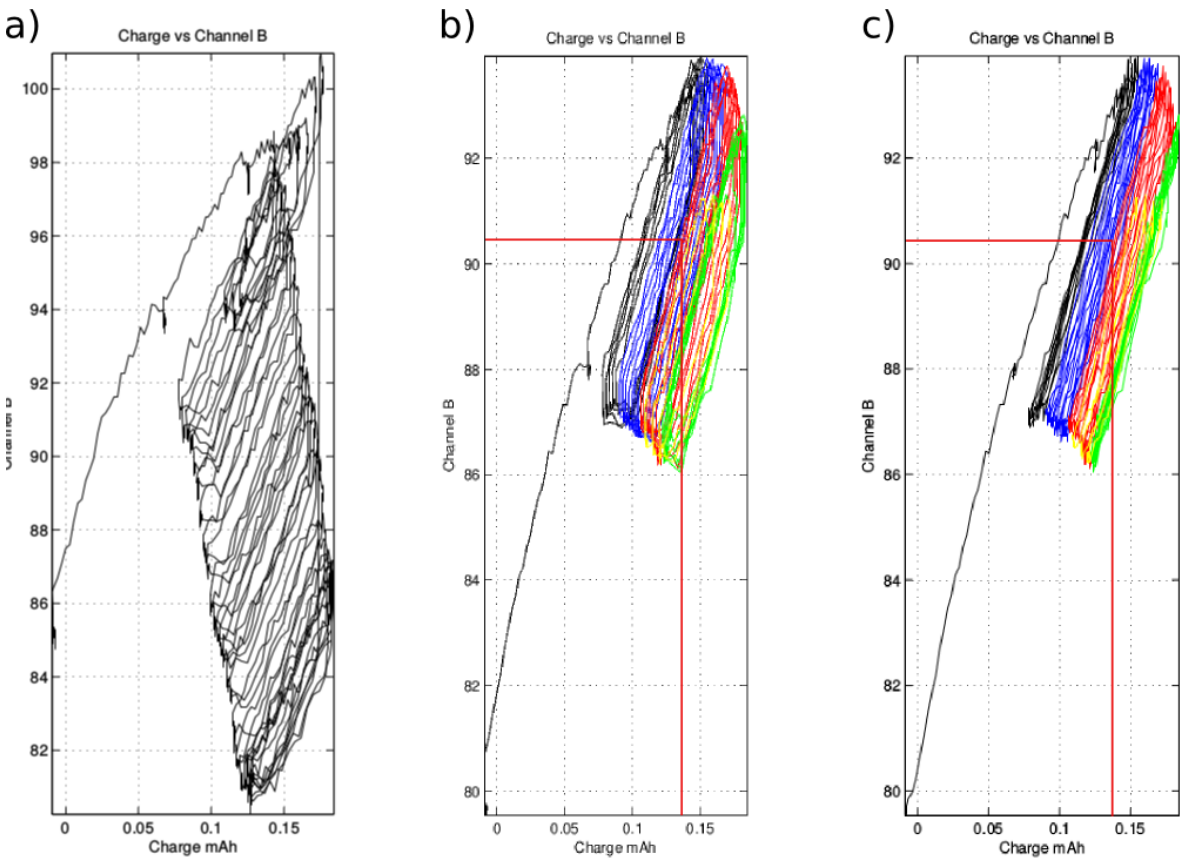


Figure 5.7.: Measurement Series 15 Cell 1 Part 2 - a) charge over intensity (channel B) without intensity compensation. b) Charge over intensity (channel B) with intensity compensation. c) charge over intensity (channel B) intensity compensated and time corrected. Both graphs shows spiral like behavior due to charing and discharging cycling.

5.3. Measurement Series - MR14

Measurement series 14 is exclusively designed to test different illumination techniques to improve color channel information from optical measurements. For this measurement series, the microscope is configured with three different illuminating LEDs, Infrared, visible and ultraviolet. Observation has been done at the same test cell (Cell 1) and identical charging/discharging cycle routine is followed.

5.3.1. Infrared Radiation Measurements

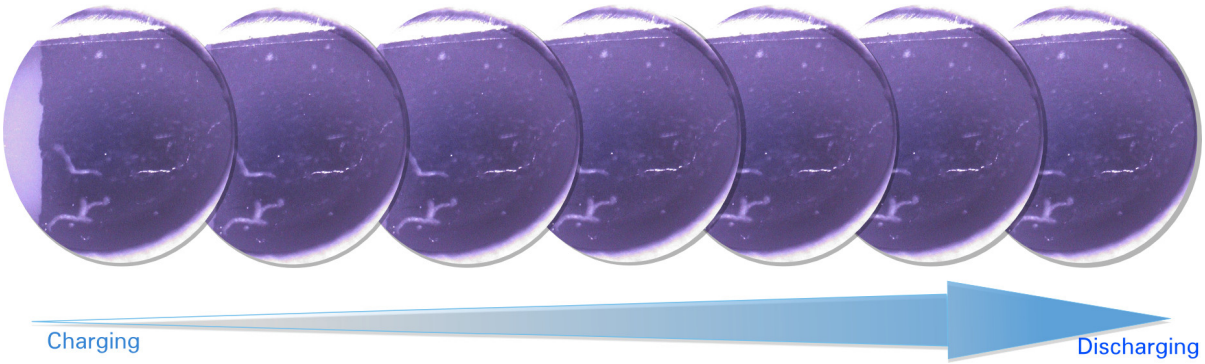


Figure 5.8.: Cathode effect: Infrared Measurements at 4th Cycle

Cell no.	Total no. Of Images	Capture frequency	Processed Images	Frequency	Note
MR14Z1P1	59700	10 sec	597	15 min per image	every 100th image

Infrared (IR) optical measurements (batch no.MR14Z1P1) include data for one week. Figure 5.8 shows microscopic view of the cell at 4th charging to discharge cycle under IR illumination. ITO and $LiFePO_4$ used as cathode materials in this instance. Optical measurements from the outlook shows almost difficult to observe any growing effects in cathode surface. This has further proved by weak color intensity behavior and histogram[4] display in figure 5.9. Even though infrared optical measurements did not improve results in this thesis, it is believed that measurements can be enhanced by using specified microscopes for infrared illumination.

[4]Histogram is a graph that shows the number of pixels in an image at each different intensity value found in that image

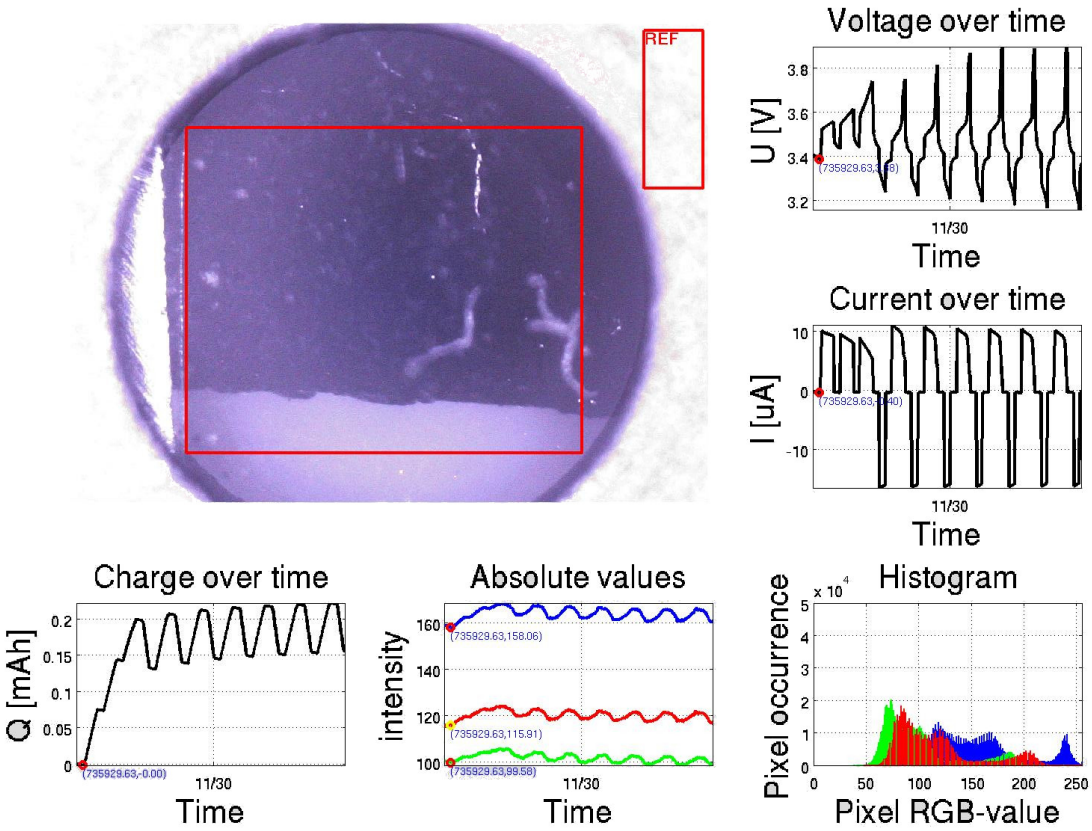


Figure 5.9.: A single frame of a video designed for electrical and infrared optical measurements. Frame contents cell voltage, current and charge as for electrical measurements. Color intensity and histogram of affected region (large rectangle in red) displayed as optical measurements.

5.3.2. Visible Radiation Measurements

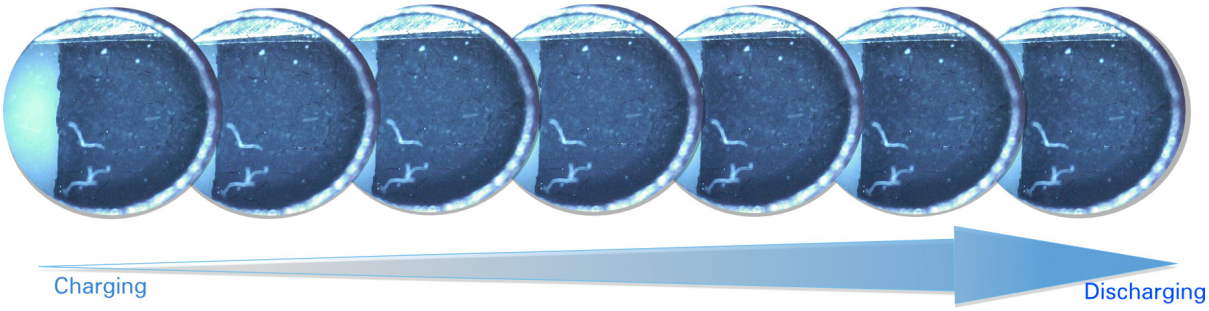


Figure 5.10.: Cathode effect: Visible radiation Measurements at 4th Cycle

Cell no.	Total no. Of Images	Capture frequency	Processed Images	Frequency	Note
MR14Z1P2	19500	10 sec	195	15 min per image	every 100th image

Visible radiation optical measurements (standard LED in camera, batch no. MR14Z1P2) include data for two days. Figure 5.10 shows microscopic view of cathode in visible light. The cathode setup is identical to the infrared measurement setup. The images are much clearer and sharper than infrared measurements and more details can be seen. However, color intensity and histogram optical measurements are quite low for image processing evaluations (see figure 5.11).

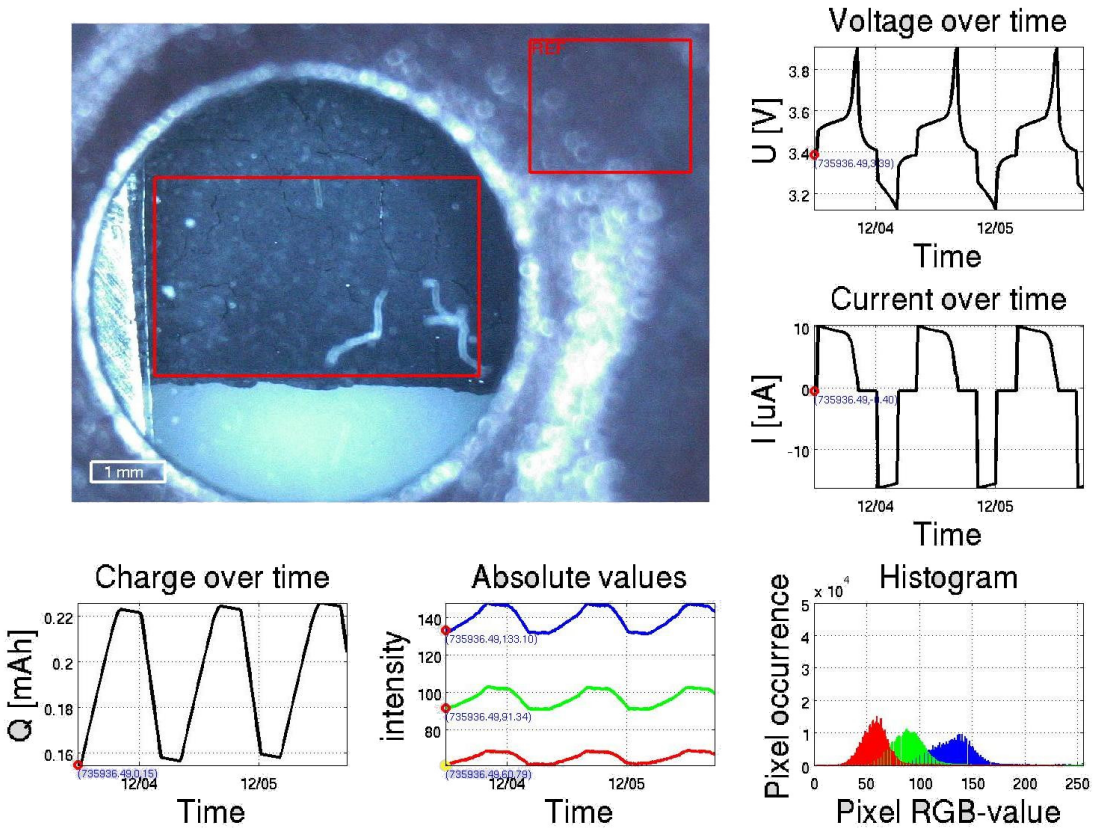


Figure 5.11.: A single frame of a video designed for electrical and visible radiation optical measurements. Frame contents cell voltage, current and charge as for electrical measurements. Color intensity and histogram of affected region (large rectangle in red) displayed as optical measurements.

5.3.3. Ultraviolet Radiation Measurements

Figure 5.12.: Cathode effect: Ultraviolet measurements at 4th cycle

Cell no.	Total no. Of Images	Capture frequency	Processed images	Frequency	Note
MR14Z1P3	58900	10 sec	589	15 min per image	every 100th image

Ultraviolet optical measurements (batch no.MR14Z1P3) include data for about one and a half days. Figure 5.12 shows microscopic view of cathode under ultraviolet radiation. Cathode setup is identical to the infrared and visible radiation measurement setup.

Optical measurements at the first glance show a difficulty to observe any growing effects in cathode surface due to high intensity of blueish color. This has further been proved by the color intensity graph which shows a constantly high blue line. The other two color intensity lines show only a very small reaction to the charge and discharge, meaning very little information on any growing effect in the cathode. Histogram information density also fairly weak (see figure 5.9).

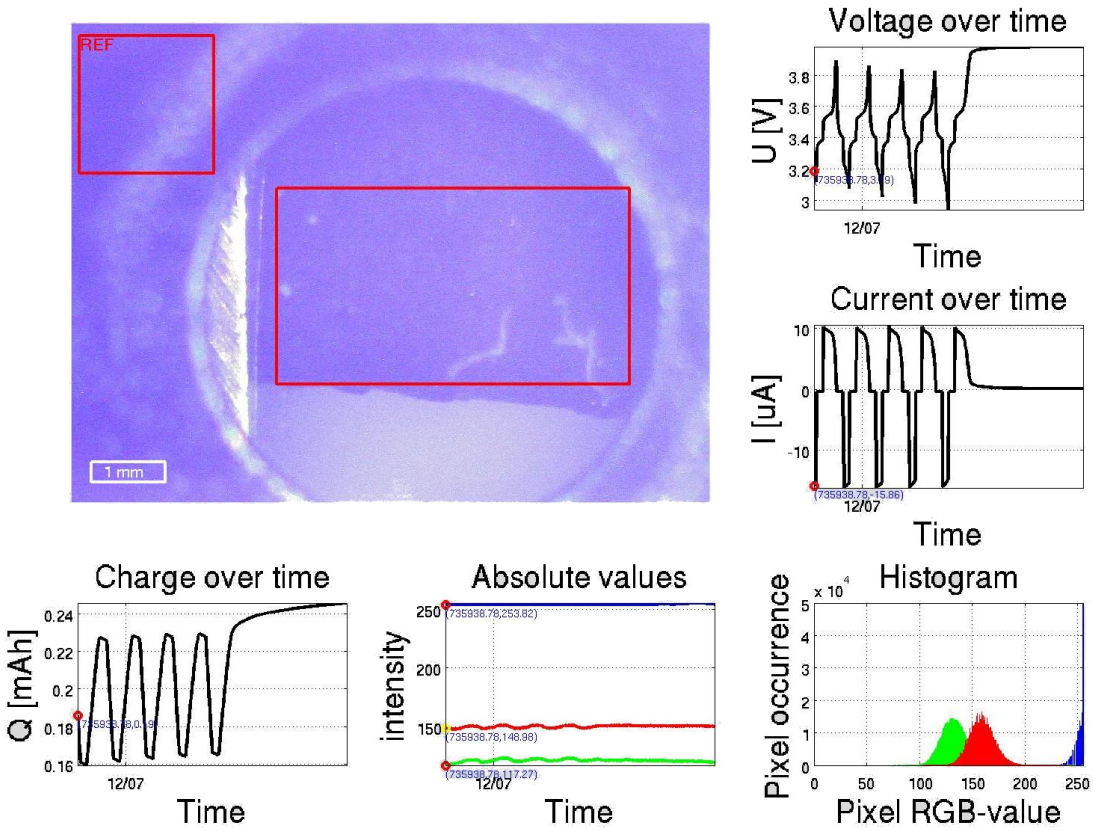


Figure 5.13.: A single frame of a video designed for electrical and ultraviolet optical measurements. Frame contents cell voltage, current and charge as for electrical measurements. Color intensity and histogram of affected region (large rectangle in red) displayed as optical measurements.

6. Conclusion

Image processing for investigation of effects in Lithium battery electrodes has produced successful results, which were evaluated in the respective chapters. The remaining open questions and ideas and concepts to improve upon the achieved results are discussed in this chapter.

6.1. Summary

This thesis presents image processing techniques that have been used to prove the correlation between electrical and chemical changes in lithium ion batteries. It is believed that these results can contribute to an accurate estimation and evaluation of the state of battery parameters such as state of charge (SoC) and battery ageing, the so called state of health (SoH).

Optically measurable chemical reactions on the anode side were firstly discovered by Harris and his research team. Taking the lead from these findings, the BATSEN project has started optical observation of the cathode of lithium-ion batteries. Test cells are utilized to model lithium-ion rechargeable batteries for the experiments. These test cells allow monitoring changes of the optical properties of cathode material through a 2 mm diameter round glass window during operation. Electrical measurements are recorded to text files whilst microscope cameras read optical measurements under different illumination into PNG image file formats. Auto exposure and white balance settings have been disabled to ensure stable and uniform color intensity levels.

Charge and discharge cycling routines for the test cells are controlled by an input command file that controls a relay board. The relay board control program, which was designed by predecessors of BATSEN team, is further modified in this thesis by integrating a loop mechanism. This minimizes the risk of errors in repetitive commands to the relay board. A new cycle plan plotter program graphically shows test cycles. Programs for further processing of measured data are created using the concept of UNIX filters 3.11.

An organized folder structure is created to store data for each measurement series for easy handling of large amounts of data (several 100s of GB optical data and several 10s of MB

electrical data as text files) B.1. It is important to preserve timestamps during data copying to the folder structures.

Various image processing methods are implemented in MATLAB scripts. Intensity compensation, binarization, edge detection and multithreshold are practiced image processing techniques in this thesis. Stacking and image subtraction are methods that are used for noise cancellation.

Intensity compensation reads Red, Green, Blue channel intensities while calibrating image brightness disarrays of the optical measurements. The results from the following technique successfully proved a correlation between cell charge and color intensities. This has further discussed in figure 5.4.

Binarization reads pixel counts of affected regions on cathode surface while masking out unwanted artifacts such as cracks, shadows, dusts etc. The results from these techniques also proved a correlation between cell charge and pixel count of the affected region (reversed polarized in phase with charge curve, see figure 5.6).

The multithreshold method is been introduced as an improved version of the image binarization method. A systematic approach is used to define threshold levels (see chapter 4.3).

Edge detection is introduced in this thesis as a way forward technique for analyze boundaries of affected regions on the cathode test cell.

Experiment results for each image processing technique are graphically visualized by producing videos for better evaluation of the results.

6.2. Discussion of Results

Measurements are carried out with various electrode materials. It is observed that the nature of chemical effects changes due to different electrode material compositions. These effects are captured by microscope every 10 sec for several weeks which generates approximately 270 000 images for each measurement series.

The cathode has been illuminated using different regions of the electromagnetic spectrum, for instance infrared radiation (IR), visible radiation and ultraviolet radiation (UV). IR and UV measurements showed weak results on all image processing techniques that have been used due to poor visibility to the affected region in the respective wavelengths.

Unexpected effects on the cathode surface such as bright spots, gas bubbles etc. hampered measurement readings on several occasions, for instance measurement series 15 and 17. The reasons for these effects are still unknown.

To enhance optical effects on the cathode during charge and discharge, different cathode material compositions have been used. Following results were observed:

Cathode material composition	Optical effect	Cell capacity
$LiFePo_4$ + Carbon	Weak Effect	high capacity
ITO(26 %) + $LiFePo_4$(65 %)	Medium effect	medium capacity
ITO(40 %) + $LiFePo_4$(50 %)	Strong effect	low capacity
Pure ITO	Very strong effect	almost 0 capacity

Table 6.1.: Optical and cell capacity dependencies on cathode material composition

In measurement series 17, a test with ITO coated glass has been conducted. In this test, the aluminium current collector is replaced by a very thin layer of optically transparent ITO on the cover glass. Then, the active battery material is applied directly on top of this coating. In this way, the cathode can be built in facing the separator, greatly increasing ion conductivity and mitigating the area effects that can be observed in previous cells. Using this technique, a uniformly distributed effect could be observed. Results are discussed in respective chapters.

Stacking and image subtraction didn't provide much improvement, but a working process has been established on which further tests can be based on.

It is acknowledged that observation on complete cathode, produce much stable intensity channel readings than observing smaller parts of the cathode. Furthermore, stable and compensated intensity color channels displays almost linear behaviour against cell charge (see figure 5.7).

6.3. Proposals for Further Improvements

The work of this thesis is very likely to be carry forwards by the BATSEN group. Image processing techniques and mass data management methods can be further examined and improved:

The synchronization of electrical and optical measurement timestamps are very important in this thesis to match optical effects on cathode surface with electrical changes, especially for video graphics and plotting. Currently, optical and electrical measurements are recorded via separate computer units running the Linux operating system. Failure of a single unit may lead to mismatch of measurements which may put the complete experiment at stake. Therefore, it would be much more reliable to have a single system for optical and electrical measurements

for each individual test cell. Regarding the camera system, more professional or specialized components should be considered.

Since the measurement setup is based in EU time zone, Daylight Saving Time (DST) is an important factor that has to be taken care of during optical and electrical measurements recording. Time adjustment has been made manually so far for daylight saving. Considering the long run, it could be better using Coordinated Universal Time (UTC) as the time base for measurements which does not require any adjustment in any occasion.

Video graphical representation and image processing tasks are mainly handled through separate MATLAB scripts in the thesis. Both these tasks have its sequential steps to be followed before it makes into a single video output. This can be time consuming and gives rise to the possibility of erroneously missing important steps. To mitigate this issue, the MATLAB script chain could be implemented in a single script. This would make fast and easy execution without having prior knowledge of the task.

Electrode preparation and cell assembly methods are largely influenced by chemical effects at the cathode surface. Gas bubble occurrence and lithium plating effects are some obstacles that measurements face due to possible mistakes occurring during cell production. This can be optimized by introducing a more sturdy cell design or automated cell production.

Bibliography

[1] *avconv Documentation*. https://libav.org/avconv.html, . – Accessed: 2015-02-28

[2] *Electropaedia State of Charge (SOC) Determination*. http://www.mpoweruk.com/soc.htm, . – Accessed: 2014-12-18

[3] *Lithium and sodium battery cathode materials: computational insights into voltage, diffusion and nanostructural properties*. http://pubs.rsc.org/en/content/articlehtml/2014/cs/c3cs60199d, . – Accessed: 2015-02-25

[4] *MATLAB Edge Detection*. http://de.mathworks.com/help/images/edge-detection.html, . – Accessed: 2014-08-18

[5] *WIKIPEDIA Bayer filter*. http://en.wikipedia.org/wiki/Bayer_filter, . – Accessed: 2015-02-18

[6] *WIKIPEDIA Binary Image*. http://en.wikipedia.org/wiki/Binary_image, . – Accessed: 2014-12-08

[7] *WIKIPEDIA Digital Imaging*. http://en.wikipedia.org/wiki/Digital_imaging, . – Accessed: 2015-02-15

[8] *WIKIPEDIA Electromagnetic spectrum*. http://commons.wikimedia.org/wiki/File:EM_spectrum.svg, . – Accessed: 2015-01-08

[9] *WIKIPEDIA Portable Network Graphics*. http://en.wikipedia.org/wiki/Portable_Network_Graphics, . – Accessed: 2015-02-18

[10] *WIKIPEDIA Pulse oximetry*. http://en.wikipedia.org/wiki/Pulse_oximetry#Function, . – Accessed: 2015-01-08

[11] *WIKIPEDIA White light sources based on wavelength converters*. http://www.ecse.rpi.edu/~schubert/Light-Emitting-Diodes-dot-org/chap21/chap21.htm, . – Accessed: 2015-02-20

[12] BÖHM, Oliver: *Java Software Engineering unter Linux*. SuSE Press, 2002. – ISBN 3–935922–53–1

[13] CAO-PAZ, A.: *A multi-point sensor based on optical fiber for the measurement of electrolyte density in lead-acid batteries*. 2011. – Accessed: 2014-11-23

[14] CONRAD: *8 Way Relay Board*. http://www.produktinfo.conrad.com/datenblaetter/175000-199999/197730-an-01-ml-8_K_RELAISKARTE_PC230V_AC16A_de_en_fr_nl.pdf. Version: 2012

[15] GMBH, EL-CELL: *User Manual: Electrochemical Test Cell ECC-Opto-Std*. http://el-cell.com/wp-content/uploads/manuals/ECC_OPTO_manual.pdf. Version: 2014

[16] GOOSSENS, Michel ; MITTELBACH, Frank ; SAMARIN, Alexander: *Der LaTeX -Begleiter*. Addison-Wesley, 2000. – ISBN 3–8273–1689–8

[17] GRIESSBACH, Jan: *Messaufbau mit Steuer- und Analysesoftware für die optische Zustandsbeobachtung von Lithiumbatterien*, Hochschule fï¿½r Angewandte Wissenschaften Hamburg, Diplomarbeit, 2014

[18] HARRIS, Stephen J.: *Direct in situ measurements of Li transport in Li-ion battery negative electrodes*. 2010. – Accessed: 2014-09-28

[19] HUNT, Andrew ; THOMAS, David: *Der Pragmatische Programmierer*. Hanser, 2003. – ISBN 3–446–22309–6

[20] KARL-RAGMAR RIEMSCHNEIDER, Matthias S.: *DRAHTLOSE SENSOREN IN DEN ZELLEN VON FAHRZEUG-BATTERIEN*. 2011. – Accessed: 2014-10-08

[21] KOHM, Markus ; MORAWKSI, Jens-Uwe: *KOMA-Script*. dante, 2003. – ISBN 3–936427–45–3

[22] LIM, Jae s.: *Two-Dimensional Signal and Image Processing*. Prentice Hall PTR, 1989. – ISBN 9–7801–3935–3222

[23] LINDEN, David ; REDDY, Thomas B.: *Linden's Handbook of Batteries,Fourth Edition*. McGraw-Hill, 2002. – ISBN 978–07–162421–X

[24] NOBUYUKI, Otsu: *A Threshold Selection Method from Gray-Level Histograms*. EEE Transactions on Systems, Man, and Cybernetics, Vol. 9. https://engineering.purdue.edu/kak/computervision/ECE661.08/OTSU_paper.pdf. Version: 1979

[25] N.PAPAMARKOS ; B.GATOS: *A New Approach for Multilevel Threshold Selection*. CVGIP: Graphical Models and Image Processing, Volume 56, Issue 5. http://www.sciencedirect.com/science/article/pii/S1049965284710339. Version: 1994

[26] SCHMATZ, Klaus-Dieter: *Java 2 Micro Edition - Entwicklung mobiler Anwendungen mit CLDC und MIDP*. dpunkt.verlag, 2004. – ISBN 3–89864–271–2

[27] STREITZ, Norbert ; NIXON, Paddy: The Disappearing Computer. In: *Communications of the ACM* 48 (2005), März, Nr. 3, S. 33–35

A. Optical Cathode Chemical Effects

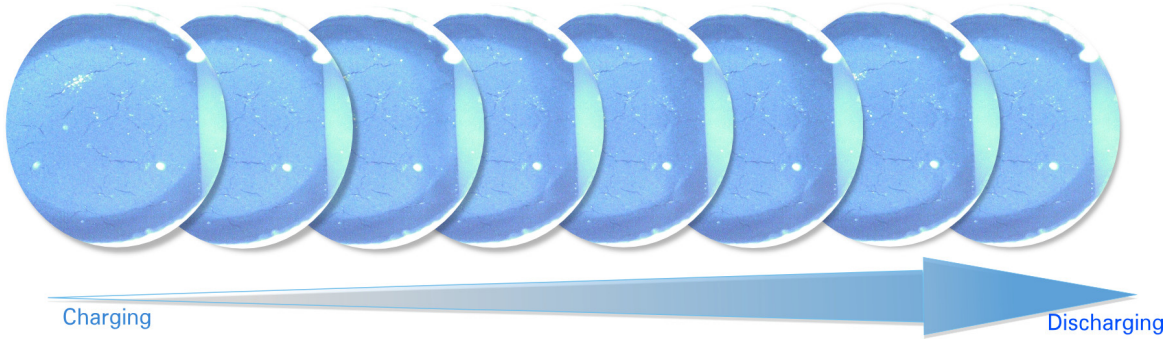


Figure A.1.: Cathode effect: Measurement series 13 at 4th Cycle

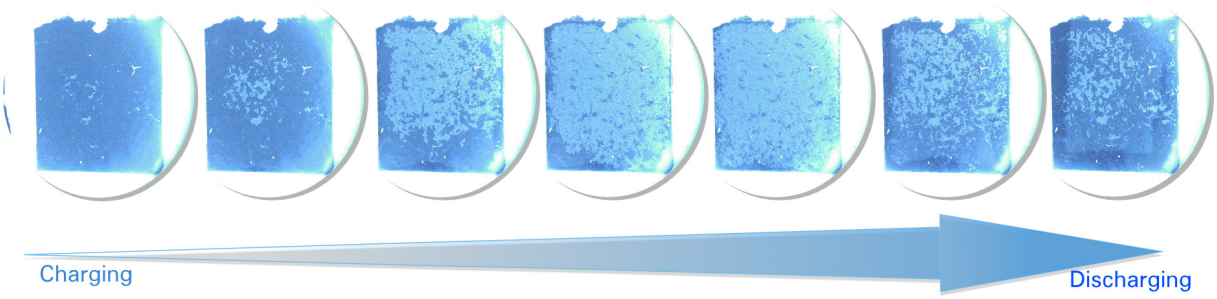


Figure A.2.: Lithium plating on cathode (measurement series 15)

Figure A.3.: Gas bubbles on cathode (measurement series 17)

B. Flow Diagrams

B.1. Folder Structure

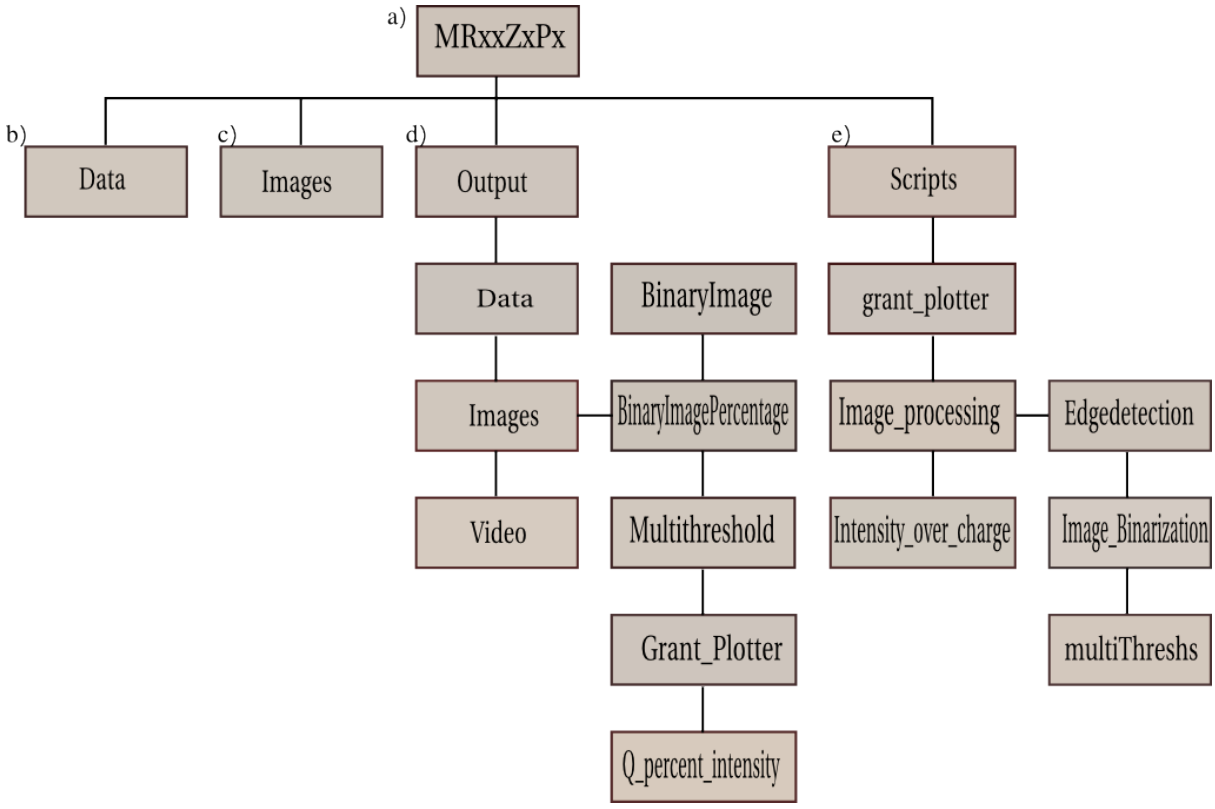


Figure B.1.: a) Electrical measurements b) Optical measurements c) Processed data including videos d) Image processing methods

B.2. Flow Diagram - countMultithreshPixels.m

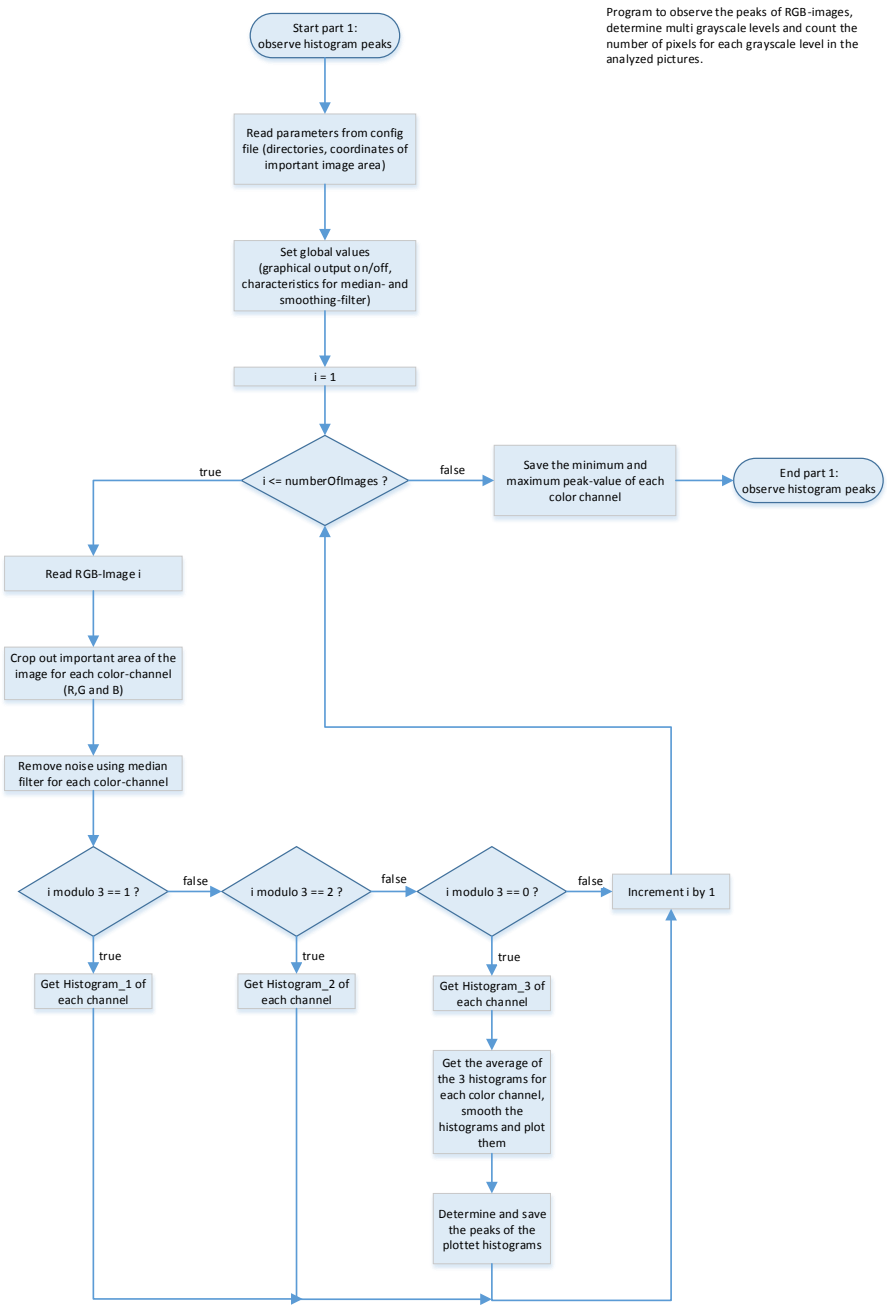


Figure B.2.: Program flow chart of Multithreshold method-**countMultithreshPixels.m**

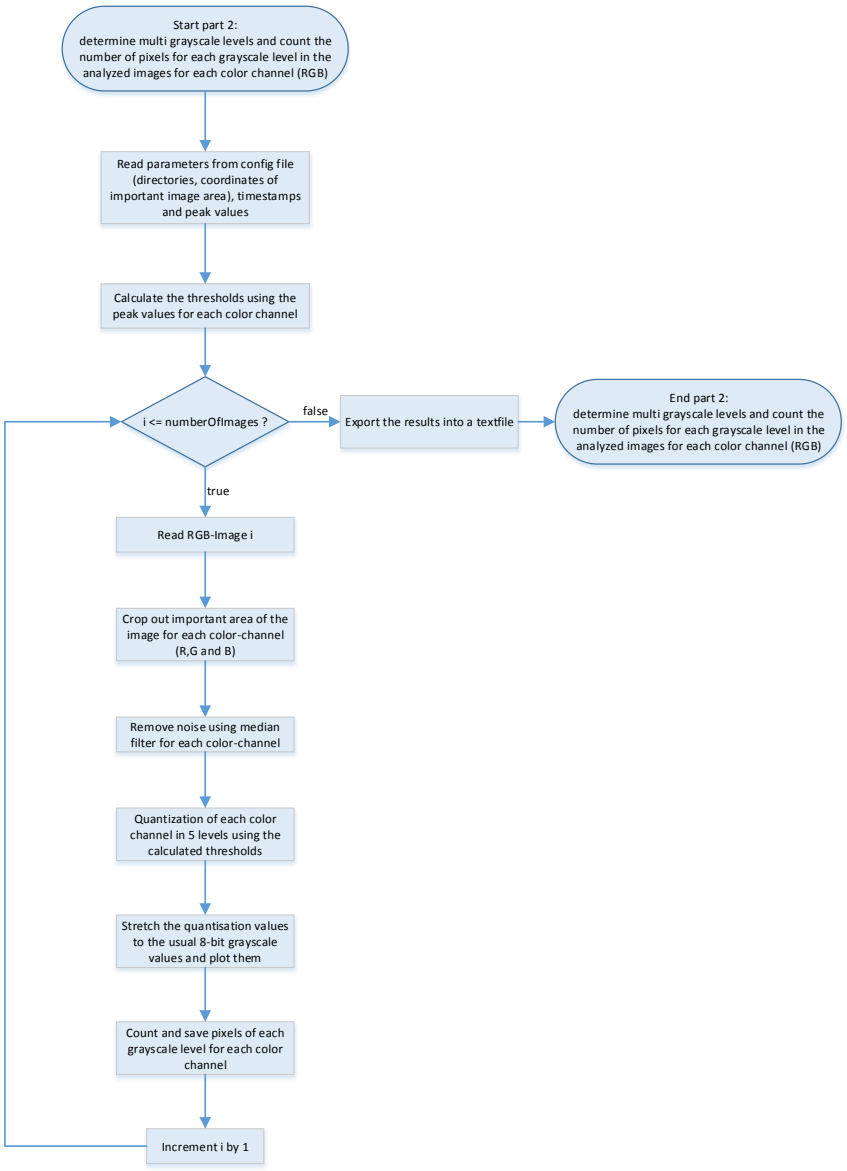


Figure B.3.: Program flow chart part 2 of Multithreshold method-**countMultithreshPixels.m**

C. Source Codes

C.1. Config.m

```
1  %%
2  % ************************Config.m**************************************
3  % author:    Don Mithila M. Palliyaguruge
4  % date:      11. Nov. 2014
5  % **********************************************************************
6
7  function param = config()
8
9  % Coordinate settings for Area of interest
10 param.xposl = 466;
11 param.xposr = 1134;
12 param.yposu = 250;
13 param.yposd = 1028;
14
15 % Coordinate settings for Reference area
16 param.ref_xposl = 24;
17 param.ref_xposr = 278;
18 param.ref_yposu = 36;
19 param.ref_yposd = 340;
20
21 % One mm in pixels (calculate from hole diameter!)
22 param.onemm = 187;
23
24 param.tempVolt = 'MR15Z1P2_Voltdata.txt';
25 param.tempCurrent = 'MR15Z1P2_Currentdata.txt';
26 param.tempCharge = 'MR15Z1P2_Chargedata.txt';
27
28 param.Outputfile = 'MR15Z1P2_RGBdata';
29 param.voltageFile = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Data/MR15Z1_spannung_20141217_2143.txt';
30 param.currentFile = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Data/MR15Z1_strom_20141217_2143_corrected.txt';
31
32 % use CurrentIntegrator to generate chargefile!
33 param.chargeFile = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Data/MR15Z1_ladung_corrected.txt';
34 param.imageSource = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Images/';
35 param.imageDest = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/MR15Z1P2-plotframe';
36
37 %Image Processing results
38 param.imageProcessingDest1 = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/BineryImage/MR15Z1P2-plotframe';
39 param.imageProcessingDest2 = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/BineryImagePercentage/MR15Z1P2-plotframe'
       ;
40
41 %charge_vs_percentage
42 param.imageProcessingDest3 = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/charge_vs_percentage/MR15Z1P2-plotframe';
43
44 %binaryImage_affectedRegion_pixel_countgraph_vidgen
45 param.imageProcessingDest6 = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/
       binaryImage_affectedRegion_pixel_countgraph_vidgen/MR15Z1P2-plotframe';
46
47 %Q_percent_intensity
48 param.imageProcessingDest4 = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/Q_percent_intensity/MR15Z1P2-plotframe';
49
50 %charge_vs_percentage_noisecheck
51 param.imageProcessingDest5 = '/media/aay339/ImageProcessing/Cells/MR15/MR15Z1P2/Output/Images/charge_vs_percentage_noisecheck/MR15Z1P2-
       plotframe';
52
53 % parameters for grant plotter
54 param.ImageRGB_data = '';
55 param.ImageArea_data = '';
```

C.2. intensitaetsverlauf_kompensiert.m

```
%% clean up
close all;
clear all;
clc;

%% this codes links to config.m file in the working directory
parser = config;

%% get the number of parts
parts      = str2num(char(inputdlg( 'Number of parts', 'Number of parts', 1, {'1'} )));

%% allocate memory for arrays
xpos       = zeros(parts, 1);
ypos       = zeros(parts, 1);
xsize      = zeros(parts, 1);
ysize      = zeros(parts, 1);
xpos_ref   = zeros(parts, 1);
ypos_ref   = zeros(parts, 1);
xsize_ref  = zeros(parts, 1);
ysize_ref  = zeros(parts, 1);
path       = cell(parts, 1);
channels   = cell(7, parts); % RGB timestamps RGB(reference)
noFiles    = zeros(parts, 1);

%% get paths and section information
for m=1:1:parts
    path(m) = cellstr(uigetdir(parser.imageSource));

    xpos(m)= parser.xposl;
    ypos(m)= parser.yposu;
    xsize(m)= parser.xposr-parser.xposl;
    ysize(m)= parser.yposd-parser.yposu;

    xpos_ref(m)  = parser.ref_xposl;
    ypos_ref(m)  = parser.ref_xposr;
    xsize_ref(m) = parser.ref_xposr - parser.ref_xposl;
    ysize_ref(m) = parser.ref_yposd - parser.ref_yposu;
    clear invalues;
end

%% process images
for m=1:1:parts
    % get all .jpg files in given directory
    files = dir(fullfile(char(path(m)), '*.png'));

    % pick number of files to be read and num of pixels
    noFiles(m)    = size(files, 1);
    noPixels      = xsize(m) * ysize(m);
    noPixels_ref = xsize_ref(m) * ysize_ref(m);

    % initialize variables
    red   = zeros(noFiles(m), 2);
    green = zeros(noFiles(m), 2);
    blue  = zeros(noFiles(m), 2);
    dates = zeros(noFiles(m), 1);
    imname= cell(noFiles(m), 1);

    % process every file
    for n=1:1:noFiles(m)
        % get image one by one and read its data
        [A, MAP, ALPHA] = imread(fullfile(char(path(m)), files(n).name), 'PNG');
        dates(n)        = files(n).datenum;
        imname(n)       = cellstr(files(n).name);

        % extract section
        section     = A(ypos(m):ypos(m)+ysize(m)-1, xpos(m):xpos(m)+xsize(m)-1, :);
        section_ref = A(ypos_ref(m):ypos_ref(m)+ysize_ref(m)-1, xpos_ref(m):xpos_ref(m)+xsize_ref(m)-1, :);
        clear A MAP ALPHA;

        % sum-up intensities on every channel
        red(n,1)   = sum(sum(section(:, :, 1))) / noPixels;
        green(n,1) = sum(sum(section(:, :, 2))) / noPixels;
        blue(n,1)  = sum(sum(section(:, :, 3))) / noPixels;
        red(n,2)   = sum(sum(section_ref(:, :, 1))) / noPixels_ref;
        green(n,2) = sum(sum(section_ref(:, :, 2))) / noPixels_ref;
        blue(n,2)  = sum(sum(section_ref(:, :, 3))) / noPixels_ref;
        clear section section_ref;

        disp(['Part: ' num2str(m) '/' num2str(parts) '   File: ', num2str(n), '/', num2str(noFiles(m))]);
    end
    channels{1, m} = red(:,1);
    channels{2, m} = green(:,1);
```

```
83      channels{3, m} = blue(:,1);
84      channels{4, m} = dates;
85      channels{5, m} = red(:,2);
86      channels{6, m} = green(:,2);
87      channels{7, m} = blue(:,2);
88      clear n red green blue dates files noPixels noPixels_ref;
89  end
90
91  % concatenate parts
92  red       = vertcat(channels{1, :});
93  green     = vertcat(channels{2, :});
94  blue      = vertcat(channels{3, :});
95  dates     = vertcat(channels{4, :});
96  red_ref   = vertcat(channels{5, :});
97  green_ref = vertcat(channels{6, :});
98  blue_ref  = vertcat(channels{7, :});
99  clear channels m;
100
101
102 %% apply correction
103 red_c=(red(:,1));
104 green_c=(green(:,1));
105 blue_c=(blue(:,1));
106 %red_c   = (red   ./ red_ref)   * mean(red_ref);
107 %green_c = (green ./ green_ref) * mean(green_ref);
108 %blue_c  = (blue  ./ blue_ref)  * mean(blue_ref);
109
110 %% output correlation
111 %disp(['Korrelation rot: ' num2str(corr(red, red_ref))]);
112 %disp(['Korrelation grï¿½n: ' num2str(corr(green, green_ref))]);
113 %disp(['Korrelation blau: ' num2str(corr(blue, blue_ref))]);
114
115 %% calculate summarized intensity
116 % sumIntensity     = (red + green + blue);
117 % sumIntensity_ref = (red_ref + green_ref + blue_ref);
118 % sumIntensity_c   = (red_c + green_c + blue_c);
119
120 %% export data to a text and a mat file
121 % [exportfile exportpath] = uiputfile;
122 exportfile = strcat(pwd, '/',parser.Outputfile);
123 udates = round(864e5 * (dates - datenum('1970', 'yyyy')))./1000-7200; % -7200: Convert to UTC
124 exportMatrix = [udates red green blue red_c green_c blue_c];
125 dlmwrite(exportfile, sprintf('#timestamp\tred (raw)\tgreen (raw)\tblue (raw)\tred (compensated)\tgreen (compensated)\tblue (compensated
        )'), 'delimiter', '');
126 dlmwrite(exportfile, exportMatrix, '-append', 'delimiter', '\t', 'precision', 15);
127 save(strcat(parser.Outputfile,'.mat'),'exportMatrix','imname');
128 clear exportpath udates exportMatrix;
```

C.3. rgb_volt_current_charge_plotter.m

```
1   clc;
2   clear all;
3
4   %% read inputdata from configplotter.m file
5   parser = config;
6
7   %% get data as a mat file
8   A = load(strcat(parser.Outputfile,'.mat'));
9   filenames = A.imname(:,1);
10
11  %% get the image time stamps to a text file
12  [status dates] = system(['sed "1{/^#/d;};s/[\t].*//g" ' pwd '/' parser.Outputfile]);
13  disp(dates);
14
15  %% save image timestamps in a temporary text file
16  fid = fopen('imageTimeStamps.txt', 'wt');
17  fprintf(fid, '%s\n', dates);
18
19  %%
20  system(['cat ' parser.voltageFile '|grep -v "#"|./TimestampMatcher -f ./imageTimeStamps.txt >' parser.tempVolt]);
21  system(['cat ' parser.currentFile '|grep -v "#"|./TimestampMatcher -f ./imageTimeStamps.txt >' parser.tempCurrent]);
22  system(['cat ' parser.chargeFile '|grep -v "#"|./TimestampMatcher -f ./imageTimeStamps.txt >' parser.tempCharge]);
23
24
25  voltMatrix    = dlmread(parser.tempVolt);
26  currentMatrix = dlmread(parser.tempCurrent);
27  chargeMatrix  = dlmread(parser.tempCharge);
28
29  delete (parser.tempVolt,parser.tempCurrent,parser.tempCharge);
30
```

```
31  %% plot
32  % try
33      unixTime = A.exportMatrix(:,1);
34      imageDate = unixTime/86400 + datenum(1970,1,1); % convert back to matlab time stamp format
35      time = (unixTime-A.exportMatrix(1,1))*1440/60; % *1440 for minutes; *86400 for seconds
36      sumIntensity_c    = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.exportMatrix(:,7));
37
38  %     fig = figure() ;
39      h = figure();
40      whitebg('w'); %change figure background color
41
42      %% ..... figure settings .....
43      %settings:
44      plotlinewidth = 1.5;
45      gridlinewidth = 1;
46
47      % font size
48  %     set(findall(h,'type','text'),'FontSize',50,'fontWeight','bold')
49
50      %Position
51      set(h,'Units','centimeters');
52      pos = get(h,'Position');
53      set(h,'PaperPositionMode','auto');
54      %set(0,'DefaultAxisFontSize',16)
55
56      %grid-line-width
57      grid on
58      set(gca,'linewidth',gridlinewidth);
59      set(gca,'GridLineStyle','-');
60
61      set(h, 'Position',[0 0 29.7 21]);
62      set(h,'PaperOrientation','landscape');
63      a = imread(strcat(parser.imageSource,char(filenames(1))));
64
65
66  %# ..... End of figure settings .....
67
68      %% plot volt graph
69
70      subplot(3,3,[1,2,4,5]);
71      imshow(a);
72
73      subplot(3,3,3);
74      y1 = voltMatrix(:,2);
75      plot(imageDate, y1, 'k','LineWidth',2);
76      datetick('x', 'keeplimits');
77      xlabel('Time','FontSize',20);
78      ylabel('U [V]','FontSize',20);
79      title('Voltage over time','FontSize',20);
80      axis tight;
81      grid on;
82
83      % create moving point + coords text
84
85      hLine1 = line('XData',imageDate(1), 'YData',y1(1), 'Color','r', ...
86          'Marker','o', 'MarkerSize',5, 'LineWidth',2);
87      hTxt1 = text(imageDate(1), y1(1), sprintf('(%.2f,%.2f)',imageDate(1),y1(1)), ...
88          'Color','b', 'FontSize',8, ...
89          'HorizontalAlignment','left', 'VerticalAlignment','top');
90
91      %% plot current graph
92
93      subplot(3,3,6);
94      y2 = currentMatrix(:,2)*1000000;
95      plot(imageDate, y2, 'k','LineWidth',2);
96      datetick('x', 'keeplimits');
97      xlabel('Time','FontSize',20);
98      ylabel('I [uA]','FontSize',20);
99      title('Current over time','FontSize',20);
100     axis tight;
101     grid on;
102
103     % create moving point + coords text
104     hLine2 = line('XData',imageDate(1), 'YData',y2(1), 'Color','r', ...
105         'Marker','o', 'MarkerSize',5, 'LineWidth',2);
106     hTxt2 = text(imageDate(1), y2(1), sprintf('(%.2f,%.2f)',imageDate(1),y2(1)), ...
107         'Color','b', 'FontSize',8, ...
108         'HorizontalAlignment','left', 'VerticalAlignment','top');
109
110     %% plot charge graph
111
112     subplot(3,3,7);
113     y3 = chargeMatrix(:,2)/3600*1000;
114     plot(imageDate,y3,'k','LineWidth',2);
115     datetick('x', 'keeplimits');
```

```
116     xlabel('Time','FontSize',20);
117     ylabel('Q_[mAh]','FontSize',20);
118     title('Charge_over_time','FontSize',20);
119     axis tight;
120     grid on;
121
122     % create moving point + coords text
123
124     hLine3 = line('XData',imageDate(1), 'YData',y3(1), 'Color','r', ...
125         'Marker','o', 'MarkerSize',5, 'LineWidth',2);
126     hTxt3 = text(imageDate(1), y3(1), sprintf('(%.2f,%.2f)',imageDate(1),y3(1)), ...
127         'Color','b', 'FontSize',8, ...
128         'HorizontalAlignment','left', 'VerticalAlignment','top');
129
130     %% plot intensity graph
131
132     subplot(3,3,8);
133     y4 = A.exportMatrix(:,5); y5 = A.exportMatrix(:,6); y6 = A.exportMatrix(:,7);
134     plot(imageDate, y4, 'r', imageDate, y5, 'g', imageDate, y6, 'b','LineWidth',2);
135     datetick('x', 'keeplimits');
136     xlabel('Time','FontSize',20);
137     ylabel('intensity','FontSize',20);
138     title('Absolute_values','FontSize',20);
139     axis tight;
140     grid on;
141
142     % create moving point + coords text
143     hLine4 = line('XData',imageDate(1), 'YData',y4(1), 'Color','y', ...
144         'Marker','o', 'MarkerSize',5, 'LineWidth',2);
145     hLine5 = line('XData',imageDate(1), 'YData',y5(1), 'Color','r', ...
146         'Marker','o', 'MarkerSize',5, 'LineWidth',2);
147     hLine6 = line('XData',imageDate(1), 'YData',y6(1), 'Color','r', ...
148         'Marker','o', 'MarkerSize',5, 'LineWidth',2);
149
150     hTxt4 = text(imageDate(1), y4(1), sprintf('(%.2f,%.2f)',imageDate(1),y4(1)), ...
151         'Color','b', 'FontSize',8, ...
152         'HorizontalAlignment','left', 'VerticalAlignment','top');
153     hTxt5 = text(imageDate(1), y5(1), sprintf('(%.2f,%.2f)',imageDate(1),y5(1)), ...
154         'Color','b', 'FontSize',8, ...
155         'HorizontalAlignment','left', 'VerticalAlignment','top');
156     hTxt6 = text(imageDate(1), y6(1), sprintf('(%.2f,%.2f)',imageDate(1),y6(1)), ...
157         'Color','b', 'FontSize',8, ...
158         'HorizontalAlignment','left', 'VerticalAlignment','top');
159
160
161     %% iterate through all images
162     for i = 1:size(A.exportMatrix, 1)    %All images
163
164         subplot(3,3,[1,2,4,5]);
165         filepath = strcat(parser.imageSource,char(filenames(i)));
166         a = imread(filepath);
167         imshow(a);
168         rectangle('Position',[parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)], 'LineWidth',2, '
                EdgeColor','r');
169         rectangle('Position',[parser.ref_xposl parser.ref_yposu (parser.ref_xposr-parser.ref_xposl) (parser.ref_yposd-parser.ref_yposu)
                ], 'LineWidth',2, 'EdgeColor','r');
170         rectangle('Position',[50 1100 parser.onemm 50], 'LineWidth',2, 'EdgeColor','w');
171         text(parser.ref_xposl+4,parser.ref_yposu+14,'REF','Color','r')
172         text(80,1117,'1_mm','Color','w')
173
174         subplot(3,3,9);
175         b(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr,3);
176         hist(double(b(:)),255);
177         hold on;
178         g(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr,2);
179
180         hist(double(g(:)),255);
181         hold on;
182         r(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr,1);
183         hist(double(r(:)),255);
184         hold off;
185         title('Histogram','FontSize',20);
186         ylabel ('Pixel_occurrence','FontSize',20);
187         xlabel ('Pixel_RGB-value','FontSize',20);
188         axis([0 255 0 50000]);
189         %axis tight;
190         grid on
191
192         h = findobj(gca,'Type','patch');
193         set(h(1),'facecolor', 'r','edgecolor','r');
194         set(h(2),'facecolor', 'g','edgecolor','g');
195         set(h(3),'facecolor', 'b','edgecolor','b');
196
197         %# update point & text
198         set(hLine1, 'XData',imageDate(i), 'YData',y1(i))
```

```
199             set(hTxt1, 'Position',[imageDate(i) y1(i)], ...
200                 'String',sprintf('(%.2f,%.2f)',[imageDate(i) y1(i)]))
201
202             set(hLine2, 'XData',imageDate(i), 'YData',y2(i))
203             set(hTxt2, 'Position',[imageDate(i) y2(i)], ...
204                 'String',sprintf('(%.2f,%.2f)',[imageDate(i) y2(i)]))
205
206             set(hLine3, 'XData',imageDate(i), 'YData',y3(i))
207             set(hTxt3, 'Position',[imageDate(i) y3(i)], ...
208                 'String',sprintf('(%.2f,%.2f)',[imageDate(i) y3(i)]))
209
210
211             set(hLine4, 'XData',imageDate(i), 'YData',y4(i))
212             set(hLine5, 'XData',imageDate(i), 'YData',y5(i))
213             set(hLine6, 'XData',imageDate(i), 'YData',y6(i))
214             set(hTxt4, 'Position',[imageDate(i) y4(i)], ...
215                 'String',sprintf('(%.2f,%.2f)',[imageDate(i) y4(i)]))
216             set(hTxt5, 'Position',[imageDate(i) y5(i)], ...
217                 'String',sprintf('(%.2f,%.2f)',[imageDate(i) y5(i)]))
218             set(hTxt6, 'Position',[imageDate(i) y6(i)], ...
219                 'String',sprintf('(%.2f,%.2f)',[imageDate(i) y6(i)]))
220
221             drawnow;                                %# force refresh
222             %pause(0.1)                            %# slow down animation
223
224             %Filename
225             numm=int2str(i);
226             set(gcf, 'InvertHardCopy', 'off'); % print to jpg exact colors of figure
227             filename = strcat(parser.imageDest,numm);
228             saveas(gcf,filename,'jpg')
229
230         end
231
232     % catch me
233     %     close;
234     %     msgbox('You have image files which does not match with volt or current data timestamps!!');
235     % end
```

C.4. binaryIm_threshold_adjuster.m

```
1   %%
2   % ***************************binaryIm_threshold_adjuster.m**********************************
3   % author:    Don Mithila M. Palliyaguruge
4   % date:      12. Feb. 2015
5   % ****************************************************************************
6
7   clc;
8   clear all;
9
10  %% read inputdata from configplotter.m file
11  parser = config;
12
13  files = dir(fullfile(char(parser.imageSource), '*.png'));
14
15  figure(1);
16
17      %% iterate through all images
18      for i = 1:size(files)    %All images
19
20          filepath = strcat(parser.imageSource,'/',files(i).name);
21          I = imread(filepath);
22          IGray_R = I(:,:,1);
23          IGray_G = I(:,:,2);
24          IGray_B = I(:,:,3);
25          IGrayCropped_R = imcrop(IGray_R, [parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)]);
26          IGrayCropped_G = imcrop(IGray_G, [parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)]);
27          IGrayCropped_B = imcrop(IGray_B, [parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)]);
28
29          subplot(5,3,[1,2,3,4,5,6]);
30          imshow(I);
31          rectangle('Position',[parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)], 'LineWidth',2, '
                EdgeColor','b');
32          title('Original image');
33
34          % calculate total number of pixel in the image
35          [rows columns] = size(IGrayCropped_R);
36          totnumOfPixels = rows*columns;
37
38
39          %.....................................R..channel.....................................................
40          % binary image
```

```
41          subplot(5,3,7);
42  %         %level = graythresh(IGrayCropped0);
43          level = 0.08;
44          binaryim = im2bw(IGrayCropped_R, level);
45          imshow(binaryim);
46          title('Channel R : Binary image');
47
48          % salt and pepper removal
49          subplot(5,3,10);
50          % B = medfilt2(binaryim,[10 10]);
51          B = medfilt2(binaryim,[20 20]);
52
53          % count dark pixels of a binary medfiltered image
54          numDarkPixels_R = sum(~B(:)); % Notice I had to invert the image with ~.
55          % calculatin percentage of the dark area against the whole image
56          percentage_R = numDarkPixels_R/totnumOfPixels * 100;
57
58          imshow(B);
59          title('Channel R : salt and pepper removed');
60          xlabel(strcat('Dark region : ',int2str(percentage_R),' %'), 'FontSize', 20);
61
62          % canny edge detector
63          subplot(5,3,13);
64          canny = edge(B,'canny',level);
65          imshow(canny);
66          title('Channel R : canny edge detector');
67          clear binaryim B canny
68          %.....G..channel.....
69                  % binary image
70          subplot(5,3,8);
71          %level = graythresh(IGrayCropped0);
72          level = 0.181;
73          binaryim = im2bw(IGrayCropped_G, level);
74          imshow(binaryim);
75          title('Channel G : Binary image');
76
77          % salt and pepper removal
78          subplot(5,3,11);
79          % B = medfilt2(binaryim,[10 10]);
80          B = medfilt2(binaryim,[20 20]);
81
82          % count dark pixels of a binary medfiltered image
83          numDarkPixels_G = sum(~B(:)); % Notice I had to invert the image with ~.
84          % calculatin percentage of the dark area against the whole image
85          percentage_G = numDarkPixels_G/totnumOfPixels * 100;
86
87          imshow(B);
88          title('Channel G : salt and pepper removed');
89          xlabel(strcat('Dark region : ',int2str(percentage_G),' %'), 'FontSize', 20);
90
91          % canny edge detector
92          subplot(5,3,14);
93          canny = edge(B,'canny',level);
94          imshow(canny);
95          title('Channel G : canny edge detector');
96          clear binaryim B canny
97          %.....B..channel.....
98                  % binary image
99          subplot(5,3,9);
100         %level = graythresh(IGrayCropped0);
101         level = 0.29;
102         binaryim = im2bw(IGrayCropped_B, level);
103         imshow(binaryim);
104         title('Channel B : Binary image');
105
106         % salt and pepper removal
107         subplot(5,3,12);
108         % B = medfilt2(binaryim,[10 10]);
109         B = medfilt2(binaryim,[20 20]);
110
111         % count dark pixels of a binary medfiltered image
112         numDarkPixels_B = sum(~B(:)); % Notice I had to invert the image with ~.
113         % calculatin percentage of the dark area against the whole image
114         percentage_B = numDarkPixels_B/totnumOfPixels * 100;
115
116         imshow(B);
117         title('Channel B : salt and pepper removed');
118         xlabel(strcat('Dark region : ',int2str(percentage_B),' %'), 'FontSize', 20);
119
120         % canny edge detector
121         subplot(5,3,15);
122         canny = edge(B,'canny',level);
123         imshow(canny);
124         title('Channel B : canny edge detector');
125         clear binaryim B canny
```

```
126
127         numm=int2str(i);
128         set(gcf, 'InvertHardCopy', 'off'); % print to jpg exact colors of figure
129         filename = strcat(parser.imageDest,numm);
130         saveas(gcf,filename,'jpg')
131
132     end
```

C.5. binaryIm_affectedRegion_pixel_percentage_filegen.m

```
1   %%
2   % ***************binaryIm_affectedRegion_pixel_percentage_filegen.m*********************************
3   % author:    Don Mithila M. Palliyaguruge
4   % date:      12. Dec. 201
5   % ************************************************************************
6
7   clc;
8   clear all;
9
10  %% read inputdata from configplotter.m file
11  parser = config;
12
13  % get all .jpg files in given directory
14  files = dir(fullfile(char(parser.imageSource), '*.png'));
15
16  % calculate commonly used values
17  noFiles    = size(files, 1);
18
19  figure(1);
20
21  %% allocate memory for arrays
22  percentage_R = zeros(noFiles,1);
23  percentage_G = zeros(noFiles,1);
24  percentage_B = zeros(noFiles,1);
25  dates = zeros(noFiles, 1);
26  udates = zeros(noFiles, 1);
27
28
29      %% iterate through all images
30      for i = 1:size(files)     %All images
31  %     for i = 1:10    %All images
32          filepath = strcat(parser.imageSource,'/',files(i).name);
33          I = imread(filepath);
34          dates(i) = files(i).datenum;
35          IGray_R = I(:,:,1);
36          IGray_G = I(:,:,2);
37          IGray_B = I(:,:,3);
38
39           % crop image
40          IGrayCropped_R = imcrop(IGray_R, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
41          IGrayCropped_G = imcrop(IGray_G, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
42          IGrayCropped_B = imcrop(IGray_B, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
43
44          % count pixels in cropped area
45          [rows columns] = size(IGrayCropped_R);
46          totnumOfPixels = rows*columns;
47
48          %.....................................R..channel.....................................................
49
50          % binary image
51          subplot(2,3,1);
52          %level = graythresh(IGrayCropped0);
53          level = 0.08;
54          binaryim = im2bw(IGrayCropped_R, level);
55          imshow(binaryim);
56          title('Channel R : Binary image');
57
58          % salt and pepper removal
59          span = 20;
60          % B = medfilt2(binaryim,[10 10]);
61          B = medfilt2(binaryim,[span span]);
62
63          % count dark pixels of a binary medfiltered image
64          numDarkPixels_R = sum(~B(:)); % Notice I had to invert the image with~.
65          % calculatin percentage of the dark area against the whole image
66          percentage_R(i) = numDarkPixels_R/totnumOfPixels * 100;
67
68          subplot(2,3,4);
69          imshow(B);
70          title('Channel R : salt and pepper removed');
71          xlabel(strcat('Dark region:  ',int2str(percentage_R(i)),' %'), 'FontSize', 20);
```

```
72
73          clear binaryim B
74
75          %.....G..channel.....
76
77          % binary image
78          subplot(2,3,2);
79          %level = graythresh(IGrayCropped0);
80          level = 0.181;
81          binaryim = im2bw(IGrayCropped_G, level);
82          imshow(binaryim);
83          title('Channel␣G␣:␣Binary␣image');
84
85          % salt and pepper removal
86          % B = medfilt2(binaryim,[10 10]);
87          B = medfilt2(binaryim,[span span]);
88
89          % count dark pixels of a binary medfiltered image
90          numDarkPixels_G = sum(~B(:)); % Notice I had to invert the image with~.
91          % calculatin percentage of the dark area against the whole image
92          percentage_G(i) = numDarkPixels_G/totnumOfPixels * 100;
93
94          subplot(2,3,5);
95          imshow(B);
96          title('Channel␣G␣:␣salt␣and␣pepper␣removed');
97          xlabel(strcat('Dark␣region:␣␣',int2str(percentage_G(i)),'␣%'), 'FontSize', 20);
98
99          clear binaryim B
100         %.....B..channel.....
101
102         % binary image
103         subplot(2,3,3);
104         %level = graythresh(IGrayCropped0);
105         level = 0.29;
106         binaryim = im2bw(IGrayCropped_B, level);
107         imshow(binaryim);
108         title('Channel␣B␣:␣Binary␣image');
109
110         % salt and pepper removal
111         % B = medfilt2(binaryim,[10 10]);
112         B = medfilt2(binaryim,[span span]);
113
114         % count dark pixels of a binary medfiltered image
115         numDarkPixels_B = sum(~B(:)); % Notice I had to invert the image with~.
116         % calculatin percentage of the dark area against the whole image
117         percentage_B(i) = numDarkPixels_B/totnumOfPixels * 100;
118
119         subplot(2,3,6);
120         imshow(B);
121         title('Channel␣B␣:␣salt␣and␣pepper␣removed');
122         xlabel(strcat('Dark␣region:␣␣',int2str(percentage_B(i)),'␣%'), 'FontSize', 20);
123
124         clear binaryim B
125         disp(['Iamge␣' int2str(i) ':␣Channel␣R␣' int2str(percentage_R(i)) '%␣Channel␣G␣' int2str(percentage_G(i)) '%␣Channel␣B␣'
                int2str(percentage_B(i)) '%']);
126     end
127
128 exportfile = strcat(pwd, '/','Area_in_percent');
129 udates = round(864e5 * (dates - datenum('1970', 'yyyy'))./1000-7200; % -7200: Convert to UTC
130 exportMatrix = [udates percentage_R percentage_G percentage_B];
131 dlmwrite(exportfile, sprintf('#timestamp␣␣␣\tChannel_R(percent.)␣\tChannel_G(percent.)␣␣\tChannel_B(percent.)␣␣'), 'delimiter', '');
132 dlmwrite(exportfile, exportMatrix,'-append', 'delimiter', '\t', 'precision', 15);
133 save(strcat('Area_in_percent','.mat'),'exportMatrix');
```

C.6. countMultithreshPixels.m

```
1  %%
2  % ***************************countMultithreshPixels.m**********************************
3  % author:    Don Mithila M. Palliyaguruge
4  % date:      12. Feb. 2015
5  % ************************************************************************
6  % Program to observe the peaks of RGB-Pictures, determine multi
7  % grayscale levels and count the number of pixels for each grayscale level
8  % in the analyzed pictures.
9  %
10 % Parameters from a config file are needed!!!
11 % ************************************************************************
12 %
13 %
14 % How to use:
15 % 1. In the directory of this file, create a folder containing the images
```

```
16  % and a subfolder named 'binaries' in it
17  % 2. Adept the paths in the config-file
18  % 3. Set the cropping frame in the config-file
19  % 4. run this file
20  % 5. results will be writtten in the files named 'multithreshOutput'
21  % (in ASCII) and 'multithreshOutput.mat' for MATLAB
22
23  % During the execution the process is shown in figures.
24  % To speed up the execution, disable the graphical output by setting the
25  % variable 'graphOut = 0'. Then  binary images will not be saved!!
26
27  %% ************************************************************************
28
29  % Step 1: find the peaks
30  clc;
31  clear all;
32  close all;
33
34  %% global settings
35  graphOut = 1;   % disable graphical output: 0   <-- no binary images will be saved
36                  % enable graphical output: 1
37
38
39  parser = config;
40  files = dir(fullfile(char(parser.imageSource), '*.png'));
41  startPic = 1;
42  endPic = size(files);
43
44
45  medfiltSize = 20;
46  span = 3;
47  spanSm = 20;
48
49
50  %% iterate through all images to find the peaks
51  for i = startPic : endPic    %All images
52
53      filepath = strcat(parser.imageSource,'/',files(i).name);
54      I = imread(filepath);
55      IGray_R = I(:,:,1);
56      IGray_G = I(:,:,2);
57      IGray_B = I(:,:,3);
58      IGrayCropped_R = imcrop(IGray_R, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
59      IGrayCropped_G = imcrop(IGray_G, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
60      IGrayCropped_B = imcrop(IGray_B, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
61
62      if graphOut == 1
63          figure(1)
64          subplot(2,1,1);
65          imshow(I);
66          rectangle('Position',[parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)], 'LineWidth',2, '
                EdgeColor','b');
67          title('Original␣image');
68          xlabel(strcat('frame␣nr:␣␣',int2str(i)), 'FontSize', 10);
69      end
70      % calculate total number of pixel in the image
71      [rows columns] = size(IGrayCropped_R);
72      totnumOfPixels = rows*columns;
73
74      %.....................................R..channel.....................................................
75
76      YR = medfilt2(IGrayCropped_R,[medfiltSize medfiltSize]); % remove noise
77      YR = imcrop(YR, [medfiltSize/2 medfiltSize/2 length(YR(1,:))-(medfiltSize) length(YR(:,1))-(medfiltSize)]);
78
79      if mod(i,span) == 1
80          [YR1_hist a] = imhist(YR);
81      end
82      if mod(i,span) == 2
83          [YR2_hist a] = imhist(YR);
84      end
85      if mod(i,span) == 0
86          [YR3_hist a] = imhist(YR);
87  %         sumR = ( smooth(YR1_hist,spanSm,'moving') + smooth(YR2_hist,spanSm,'moving') + smooth(YR3_hist,spanSm,'moving'));
88          sumR = ( YR1_hist + YR2_hist + YR3_hist);
89          YR_hist(i/span,:) = sumR/3;
90
91          [peak,locR(i/span)] = max(YR_hist(i/span,:));
92          if graphOut == 1
93              subplot(2,1,2);
94              plot((0:255),YR_hist(i/span,:),'r')
95              hold on
96              %ylim([0 9000])
97          end
98      end
99
```

```
100     %.....G..channel.....
101
102     YG = medfilt2(IGrayCropped_G,[medfiltSize medfiltSize]); % remove noise
103     YG = imcrop(YG, [medfiltSize/2 medfiltSize/2 length(YG(1,:))-(medfiltSize) length(YG(:,1))-(medfiltSize)]);
104
105     if mod(i,span) == 1
106         [YG1_hist a] = imhist(YG);
107     end
108     if mod(i,span) == 2
109         [YG2_hist a] = imhist(YG);
110     end
111     if mod(i,span) == 0
112         [YG3_hist a] = imhist(YG);
113         sumG = ( smooth(YG1_hist,spanSm,'moving') + smooth(YG2_hist,spanSm,'moving') + smooth(YG3_hist,spanSm,'moving'))/3;
114         YG_hist(i/span,:) = sumG;
115
116         [peak,locG(i/span)] = max(YG_hist(i/span,:));
117         if graphOut == 1
118             subplot(2,1,2);
119             plot((0:255),YG_hist(i/span,:),'g')
120             hold on
121             grid on
122         end
123     end
124
125     %.....B..channel.....
126
127     YB = medfilt2(IGrayCropped_B,[medfiltSize medfiltSize]); % remove noise
128     YB = imcrop(YB, [medfiltSize/2 medfiltSize/2 length(YB(1,:))-(medfiltSize) length(YB(:,1))-(medfiltSize)]);
129
130     if mod(i,span) == 1
131         [YB1_hist a] = imhist(YB);
132     end
133     if mod(i,span) == 2
134         [YB2_hist a] = imhist(YB);
135     end
136     if mod(i,span) == 0
137         [YB3_hist a] = imhist(YB);
138         sumB = ( smooth(YB1_hist,spanSm,'moving') + smooth(YB2_hist,spanSm,'moving') + smooth(YB3_hist,spanSm,'moving'))/3;
139         YB_hist(i/span,:) = sumB;
140
141         [peak,locB(i/span)] = max(YB_hist(i/span,:));
142         if graphOut == 1
143             subplot(2,1,2);
144             plot((0:255),YB_hist(i/span,:),'b')
145             title({['Rmin:', num2str( min(locR)),';␣Rmax␣:', num2str( max(locR))]; ['Gmin:', num2str( min(locG)),'␣Gmax:', num2str( max
                    (locG))]; ['Bmin:', num2str( min(locB)),'␣Bmax:', num2str( max(locB))] });
146         end
147     end
148 %     if i==1
149 %         pause;
150 %     end
151 end
152 Rmin=min(locR);
153 Rmax=max(locR);
154 Gmin=min(locG);
155 Gmax=max(locG);
156 Bmin=min(locB);
157 Bmax=max(locB);
158
159 csvwrite('peakVals.dat',[Rmin Rmax Gmin Gmax Bmin Bmax])
160
161 pause(1);
162
163
164
165 %% ****************************************************************
166 % Step 2: determine the multi grayscale levels and count pixel
167 % clc;
168 % clear all;
169
170 %% read inputdata from configplotter.m file
171 % parser = config_MR13Z3;
172
173 unixTime     = dlmread('imageTimeStamps.txt'); % read time duration from imageTimeStamps text file
174 imageDate = unixTime/86400 + datenum(1970,1,1); % convert back to matlab time stamp format
175
176 %% global settings
177 numberOfThreshs = 4; % only for auto scaled thresholds; number of thresholds will provide N+1 Grayscalevalues
178 medfiltSize = 20; % m x m-size for 'medfilt2' (also used to crop the black frame from filtering
179
180 % thresholds
181 M = csvread('peakVals.dat');
182 cor = 0;
183 Rmin = M(1)-cor;
```

```
184 Rmax = M(2)-cor;
185 Gmin = M(3)-cor;
186 Gmax = M(4)-cor;
187 Bmin = M(5)-cor;
188 Bmax = M(6)-cor;
189
190 %% allocate memory for arrays
191 fileIndex = 1 : size(files);
192 percentage_R = zeros(size(imageDate,1),1);
193 percentage_G = zeros(size(imageDate,1),1);
194 percentage_B = zeros(size(imageDate,1),1);
195
196 files = dir(fullfile(char(parser.imageSource), '*.png'));
197
198 if graphOut == 1
199     figure(2);
200 end
201 % subplot(6,3,16);
202 % plot(imageDate,percentage_R);
203 % datetick('x', 'dd.mm.yy');
204 % xlabel('Time'); ylabel('Percentage');
205 % title('Area plot in percentage');
206 % ylim([0 100]);
207 % axis tight;
208 % grid on;
209
210 darkSum = 0;
211
212 % Matrix for the number of the grayscalevalues of each color channel
213 h=size(files);
214 numberOfGrayValues = zeros(h(1),15);
215 % udates = zeros(h(1),1);
216 dates = zeros(h(1),1);
217
218
219
220 threshs_R =[Rmin  Rmin+(Rmax-Rmin)/3  Rmax-(Rmax-Rmin)/3 Rmax];
221 threshs_G =[Gmin  Gmin+(Gmax-Gmin)/3  Gmax-(Gmax-Gmin)/3 Gmax];
222 threshs_B =[Bmin  Bmin+(Bmax-Bmin)/3  Bmax-(Bmax-Bmin)/3 Bmax];
223
224
225 %% iterate through all images
226 for i = startPic : endPic%size(files)    %All images
227   %%
228     dates(i) = files(i).datenum;
229
230     filepath = strcat(parser.imageSource,'/',files(i).name);
231     I = imread(filepath);
232     IGray_R = I(:,:,1);
233     IGray_G = I(:,:,2);
234     IGray_B = I(:,:,3);
235     IGrayCropped_R = imcrop(IGray_R, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
236     IGrayCropped_G = imcrop(IGray_G, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
237     IGrayCropped_B = imcrop(IGray_B, [parser.xposl parser.yposu parser.xposr-parser.xposl parser.yposd-parser.yposu]);
238
239     if graphOut == 1
240         subplot(2,3,[1,2,3]);
241         imshow(I);
242         rectangle('Position',[parser.xposl parser.yposu (parser.xposr-parser.xposl) (parser.yposd-parser.yposu)], 'LineWidth',2, '
                EdgeColor','b');
243         title('Original image');
244         xlabel(strcat('frame nr: ',int2str(i)), 'FontSize', 10);
245     end
246     % calculate total number of pixel in the image
247     [rows columns] = size(IGrayCropped_R);
248     totnumOfPixels = rows*columns;
249
250
251     %.....R..channel.....
252
253     %---------- MULTITHRESHOLDING FROM HERE----------
254     YR = medfilt2(IGrayCropped_R,[medfiltSize medfiltSize]); % remove noise
255     %         if i==33
256     %         Y_test4 = medfilt2(IGrayCropped_R,[medfiltSize medfiltSize]); % remove noise
257     %         Y_test4 = imcrop(Y_test4, [medfiltSize/2 medfiltSize/2 length(Y_test4(1,:))-(medfiltSize) length(Y_test4(:,1))-(
            medfiltSize)]);
258     %         threshs =[60  68.75  77.5 86.25]; % used nr 44
259     %         seg_I4 = imquantize(Y_test4,threshs);
260     %         end;
261     %         if i==5
262     %         Y_test5 = medfilt2(IGrayCropped_R,[medfiltSize medfiltSize]); % remove noise
263     %         Y_test5 = imcrop(Y_test5, [medfiltSize/2 medfiltSize/2 length(Y_test5(1,:))-(medfiltSize) length(Y_test5(:,1))-(
            medfiltSize)]);
264     %         threshs =[60  68.75  77.5 86.25]; % used nr 44
265     %         seg_I5 = imquantize(Y_test5,threshs);
```

```
266     %         end;
267     % cut off the black frame from filtering
268     YR = imcrop(YR, [medfiltSize/2 medfiltSize/2 length(YR(1,:))-(medfiltSize) length(YR(:,1))-(medfiltSize)]);
269     % get reference-thresholds
270     % threshs = multithresh(Y,numberOfThreshs); % calculates multi thresholds
271     %         if i==44
272     %             threshR = threshs
273     %         end
274
275     seg_IR = imquantize(YR,threshs_R);  % segmentation writes 1,2,3,...,Number of Graylevels to the image array
276
277     seg_IR_scaled = (seg_IR-1).*(255/numberOfThreshs); % scale the segment-values (1,..5) ti grayscalevalues (0,...255)
278
279     % plot multithreshold image
280     if graphOut == 1
281         subplot(2,3,4);
282         imshow(uint8(seg_IR_scaled));
283         title(['Channel_R_:', num2str( numberOfThreshs+1 ),'_Gray-levels']);
284     end
285
286     % count pixel of each grayscale level
287     listR = imhist(uint8(seg_IR_scaled));
288     numberOfGrayValues(i,1) = listR(1);
289     numberOfGrayValues(i,2) = listR(65);
290     numberOfGrayValues(i,3) = listR(129);
291     numberOfGrayValues(i,4) = listR(192);
292     numberOfGrayValues(i,5) = listR(256);
293
294     %------------ END PART OF MULTITHRESHOLDING
295
296
297
298     %.....G..channel.....
299
300     %--------- MULTITHRESHOLDING FROM HERE-----------------------
301     YG = medfilt2(IGrayCropped_G,[medfiltSize medfiltSize]); % remove noise
302     % cut off the black frame from filtering
303     YG = imcrop(YG, [medfiltSize/2 medfiltSize/2 length(YG(1,:))-(medfiltSize) length(YG(:,1))-(medfiltSize)]);
304     % get reference-thresholds
305     % threshs = multithresh(Y,numberOfThreshs); % calculates multi thresholds
306     %         if i==44
307     %             threshG = threshs
308     %         end
309     seg_IG = imquantize(YG,threshs_G);  % segmentation writes 1,2,3,...,Number of Graylevels to the image array
310     seg_IG_scaled = (seg_IG-1).*(255/numberOfThreshs); % scale the segment-values (1,..5) to grayscalevalues (0,...255)
311
312
313     % plot multithreshold image
314     if graphOut == 1
315         subplot(2,3,5);
316         imshow(uint8(seg_IG_scaled));
317         title(['Channel_G_:', num2str( numberOfThreshs+1 ),'_Gray-levels']);
318     end
319
320     % count pixel of each grayscale level
321     listG = imhist(uint8(seg_IG_scaled));
322     numberOfGrayValues(i,6) = listG(1);
323     numberOfGrayValues(i,7) = listG(65);
324     numberOfGrayValues(i,8) = listG(129);
325     numberOfGrayValues(i,9) = listG(192);
326     numberOfGrayValues(i,10) = listG(256);
327
328     %------------ END PART OF MULTITHRESHOLDING
329
330     %.....B..channel.....
331
332     %--------- MULTITHRESHOLDING FROM HERE-----------------------
333     YB = medfilt2(IGrayCropped_B,[medfiltSize medfiltSize]); % remove noise
334     % cut off the black frame from filtering
335     YB = imcrop(YB, [medfiltSize/2 medfiltSize/2 length(YB(1,:))-(medfiltSize) length(YB(:,1))-(medfiltSize)]);
336     % get reference-thresholds
337     threshs = multithresh(YB,numberOfThreshs); % calculates multi thresholds
338     %         if i==44
339     %             threshB = threshs
340     %         end
341     seg_IB = imquantize(YB,threshs_B);  % segmentation writes 1,2,3,...,Number of Graylevels to the image array
342     seg_IB_scaled = (seg_IB-1).*(255/numberOfThreshs); % scale the segment-values (1,..5) ti grayscalevalues (0,...255)
343
344     if graphOut == 1
345         subplot(2,3,6);
346         % plot multithreshold image
347         imshow(uint8(seg_IB_scaled));
348         title(['Channel_B_:', num2str( numberOfThreshs+1 ),'_Gray-levels']);
349     end
350
```

```
351         % count pixel of each grayscale level
352         listB = imhist(uint8(seg_IB_scaled));
353         numberOfGrayValues(i,11) = listB(1);
354         numberOfGrayValues(i,12) = listB(65);
355         numberOfGrayValues(i,13) = listB(129);
356         numberOfGrayValues(i,14) = listB(192);
357         numberOfGrayValues(i,15) = listB(256);
358
359         %------------ END PART OF MULTITHRESHOLDING----------------------
360
361
362         numm=int2str(i);
363
364         if graphOut == 1
365         set(gcf, 'InvertHardCopy', 'off'); % print to jpg
366         filename = strcat(parser.imageDest,numm);
367         saveas(gcf,filename,'jpg')
368         end
369     %%
370     %       if i==1
371     %           pause;
372     %       end
373
374     end
375
376     text = {['%%**************************************************************************'];...
377         ['% This file was created with "',mfilename, '.m", written by Don Mithila M. Palliyaguruge'];...
378         ['% creation date: ', date];...
379         ['%**************************************************************************'];...
380         ['%  '];...
381         ['% Description:'];...
382         ['% during the first step, the program in "',mfilename, '.m" searches the peaks in '];...
383         ['% the histograms of the R, G and B-channel for each picture. With the iteration '];...
384         ['% through all pictures, the peaks are moving related to the color changes '];...
385         ['% during charge/discharge processes of the analyzed cell.'];...
386         ['% The Values from the peak search are used in the second step of the program, '];...
387         ['% to calculate the multi grayscale value thresholds for all pictures.'];...
388         ['%  '];...
389         ['% *For each grayscale-level the number of pixels are counted and listed below!*'];...
390         ['%**************************************************************************'];...
391         ['%  '];...
392         ['% Name of the first analyzed file:    ', files(1).name];...
393         ['%  '];...
394         ['% The values are listet in the following order:'];...
395         ['% index    timestamp    Red_GrayLvl_0    Red_GrayLvl_64    Red_GrayLvl_128    Red_GrayLvl_191    Red_GrayLvl_255    '
                Green_GrayLvl_0    Green_GrayLvl_64    Green_GrayLvl_128    Green_GrayLvl_191    Green_GrayLvl_255    '
                Blue_GrayLvl_0    Blue_GrayLvl_64 Blue_GrayLvl_128    Blue_GrayLvl_191    Blue_GrayLvl_255 ']};
396
397     text = text';
398     txt=sprintf([repmat('%s\t',1,size(text,1)),'\n'],text{:});
399     %% save data to .mat and .txt
400     % exportfile = strcat(pwd, '/',parser.Outputfile);
401     filenameExport = 'multithreshOutput';
402     udates = round(864e5 * (dates - datenum('1970', 'yyyy'))./1000-7200; % -7200: Convert to UTC
403     udates(udates == -62167312800) = 0;
404     exportMatrix = [fileIndex' udates numberOfGrayValues];
405     dlmwrite(filenameExport, txt, 'delimiter', '');
406     dlmwrite(filenameExport, exportMatrix, '-append', 'delimiter', '\t', 'precision', '%6.0f');
407     save(strcat(filenameExport,'.mat'),'exportMatrix');
408     clear exportpath udates exportMatrix;
```

C.7. relaisCardController.c

```
1   /*****************************************************************
2    * RelaisCard controller (Conrad 197730)
3    *
4    * File:     relaisCardController.c
5      Version: 0.1
6    * Date:     2014-04-18
7    * First Author:  Jan Griessbach
8
9      Version: 0.2
10   * Date:     2014-11-10
11   * Second Author:  Don Mithila M. Palliyaguruge
12   *
13   *****************************************************************/
14
15  #include <stdio.h>
16  #include <stdlib.h>
17  #include <unistd.h>
18  #include <termios.h>
```

```
19  #include <sys/select.h>
20  #include <string.h>
21  #include <fcntl.h>
22  #include <errno.h>
23  #include <time.h>
24  #include <sys/time.h>
25
26  #define TRUE  1
27  #define FALSE 0
28
29  #define READTIMEOUTUSEC 25000L /* 25ms */
30  #define FRAMELENGTH     4
31  #define BUFFERSIZE      100
32
33  #define CMD_NOOPERATION 0
34  #define CMD_SETUP       1
35  #define CMD_GETPORT     2
36  #define CMD_SETPORT     3
37  #define CMD_GETOPTION   4
38  #define CMD_SETOPTION   5
39  #define CMD_SETSINGLE   6
40  #define CMD_DELSINGLE   7
41  #define CMD_TOGGLE      8
42
43
44  struct str_frame
45  {
46    unsigned char command;
47    unsigned char address;
48    unsigned char data;
49    unsigned char checksum;
50  };
51
52  typedef struct str_stack_element
53  {
54    unsigned int    iteration;
55    long            position;
56    int             line;
57    struct str_stack_element *next;
58  } STACK_ELEMENT;
59
60  typedef union un_frame
61  {
62    unsigned char     buffer[FRAMELENGTH];
63    struct str_frame  frame;
64  } FRAME;
65
66  void usage(char *progname) {
67   fprintf(stderr, "\tUSAGE:\t %s -f commandfile -d device\n"
68                   "\n\tOPTIONS:"
69                       " -f\tfile containing control commands\n"
70                   "\t\t -d\tSerial port device\n",
71           progname
72   );
73  }
74
75  void write_frame(int dev, FRAME *frame)
76  {
77    write(dev, frame->buffer, FRAMELENGTH * sizeof(unsigned char));
78    tcflush(dev, TCOFLUSH); /* force the data to be sent */
79  }
80
81  int read_frame(int dev, FRAME *frame)
82  {
83    int             retval;
84    struct timeval  readtimeout;
85    fd_set          readset;
86
87    /* initialize filedescriptor sets and timeout structure */
88    FD_ZERO(&readset);
89    FD_SET(dev, &readset);
90    readtimeout.tv_sec  = 0L;
91    readtimeout.tv_usec = READTIMEOUTUSEC;
92
93    /* wait for input data */
94    retval = select(dev+1, &readset, NULL, NULL, &readtimeout);
95    if( retval < 0 ) {
96      fprintf(stderr, "# ERROR: %s\n", strerror(errno));
97      return -1;
98    }
99    if( retval == 0 ) {
100     fprintf(stderr, "Relaiscard did not respond in time\n");
101     return -1;
102   }
103
```

```
104     /* read return value from card */
105     read(dev, frame->buffer, FRAMELENGTH);
106     return 1;
107   }
108
109   unsigned char calculate_checksum(const FRAME *frame)
110   {
111     return (frame->frame.command ^ frame->frame.address ^ frame->frame.data);
112   }
113
114   void init_frame(FRAME *frame, unsigned char command, unsigned char address, unsigned char data)
115   {
116     frame->frame.address  = address;
117     frame->frame.command  = command;
118     frame->frame.data     = data;
119     frame->frame.checksum = calculate_checksum(frame);
120   }
121
122   int verifiy_frame(const FRAME *frame)
123   {
124     return ( calculate_checksum(frame) == frame->frame.checksum ) ? TRUE : FALSE;
125   }
126
127   void place_command(int dev, FRAME *frame)
128   {
129     FRAME answer;
130     int result;
131
132     tcflush(dev, TCIFLUSH);
133     write_frame(dev, frame);
134     result = read_frame(dev, &answer);
135
136     while(   (result < 0)                                                    /* read failed */
137           || !verifiy_frame(&answer)                                         /* frame corrupt */
138           || (answer.frame.command != (unsigned char)~(frame->frame.command)) /* invalid answer */
139          ) {
140       fprintf(stderr, "Command␣failed,␣retrying\n");
141       tcflush(dev, TCIFLUSH);
142       write_frame(dev, frame);
143       result = read_frame(dev, &answer);
144     }
145   }
146
147   int main(int argc, char *argv[])
148   {
149     int             ser_fd, opt;
150     FILE            *cmd_fp;
151     char            *cmdfilename, *devicename, linebuffer[100];
152     int             mandatoryOptions = 2;
153     struct termios  current_settings, old_settings;
154     FRAME           myframe;
155     int             linecnt = 0;
156     STACK_ELEMENT   *root_element = NULL;
157
158     /* evaluate commandline options */
159     while( (opt = getopt(argc, argv, "f:d:")) != -1 ) {
160       switch( opt ) {
161         case 'f': cmdfilename = optarg;
162                   mandatoryOptions--;
163         break;
164         case 'd': devicename = optarg;
165                   mandatoryOptions--;
166         break;
167         default:  usage(argv[0]);
168                   exit(EXIT_FAILURE);
169       }
170     }
171
172     if(mandatoryOptions > 0) {
173       usage(argv[0]);
174       exit(EXIT_FAILURE);
175     }
176
177     /* initialize termios structure */
178     memset(&current_settings, 0x00, sizeof(current_settings));
179     memset(&current_settings, 0x00, sizeof(old_settings));
180     current_settings.c_cflag     = CLOCAL | CS8 | CREAD;
181     current_settings.c_iflag     = IGNPAR;
182     current_settings.c_oflag     = 0;
183     current_settings.c_lflag     = 0;
184     current_settings.c_cc[VMIN]  = 4;
185     current_settings.c_cc[VTIME] = 0;
186     cfsetispeed(&current_settings, B19200);
187     cfsetospeed(&current_settings, B19200);
188
```

```
189     /* open serial port */
190     if( (ser_fd = open(devicename, O_RDWR)) < 0 ) {
191       fprintf(stderr, "Can't open serial port.\n");
192       return EXIT_FAILURE;
193     }
194
195     /* open commandfile */
196     if( (cmd_fp = fopen(cmdfilename, "r")) == NULL ) {
197       fprintf(stderr, "Can't open commandfile.\n");
198       close(ser_fd);
199       return EXIT_FAILURE;
200     }
201
202     /* save current serial port settings and apply new settings */
203     tcgetattr(ser_fd, &old_settings);
204     tcsetattr(ser_fd, TCSANOW, &current_settings);
205
206     /* initialize cards */
207     memset(&myframe, 0x00, sizeof(myframe));
208     init_frame(&myframe, CMD_SETUP, 1, 0);
209     place_command(ser_fd, &myframe);
210     sleep(1); /* initialization may take a while, thus wait a second */
211
212     /* process the command file */
213     while(NULL != fgets(linebuffer, BUFFERSIZE, cmd_fp)) {
214       int address, data, waittime = 0;
215       struct tm       *timestruct;
216       struct timeval timestamp;
217       char timestring[50];
218
219       linecnt++;
220
221       /* check if line is a comment */
222       if( linebuffer[0] == '#' )
223         continue;
224
225       /* check if line is a wait statement */
226       if( 0 == strncasecmp(linebuffer, "wait ", 5)) {
227           if( 1 != sscanf(linebuffer+5, "%d\n", &waittime) ) {
228             fprintf(stderr, "Syntax error on line %d.\n", linecnt);
229             continue;
230           }
231           gettimeofday(&timestamp, NULL); //get the system time and put in var timestamp(UTC)
232           timestamp.tv_sec += waittime;
233           timestruct = localtime(&(timestamp.tv_sec));
234           strftime(timestring, 50, "Waiting until %Y-%m-%d %H:%M:%S.", timestruct);
235           puts(timestring);
236           sleep(waittime);
237           continue;
238       }
239
240       /* check if line is a setport statement */
241       if( 0 == strncasecmp(linebuffer, "setport ", 8)) {
242         if( 2 != sscanf(linebuffer+8, "%d %d\n", &address, &data) ) {
243           fprintf(stderr, "Syntax error on line %d.\n", linecnt);
244           continue;
245         }
246         init_frame(&myframe, CMD_SETPORT, address, data);
247         place_command(ser_fd, &myframe);
248         continue;
249       }
250
251      /* check if line is a loop statement */
252      if( 0 == strncasecmp(linebuffer, "loop ", 5)) {
253        STACK_ELEMENT *se;
254        se = (STACK_ELEMENT*) malloc(sizeof(STACK_ELEMENT));
255        if(se == NULL){
256          fprintf(stderr, "Out of memory error!\n");
257          return EXIT_FAILURE;
258        }
259
260        if( 1 != sscanf(linebuffer+5, "%d\n", &(se->iteration)) ) {
261          fprintf(stderr, "Syntax error on line %d.\n", linecnt);
262          continue;
263        }
264
265        se->position = ftell(cmd_fp);
266        se->line     = linecnt;
267        se->next = root_element;
268        root_element = se;
269        continue;
270      }
271
272       /* check if line is a endloop statement */
273       if( 0 == strncasecmp(linebuffer, "endloop", 7)) {
```

```
274         if(NULL == root_element) {
275           fprintf(stderr, "Syntax␣error␣on␣line␣%d.␣endloop␣command␣without␣a␣prior␣loop␣command\n", linecnt);
276           continue;
277         }
278
279         if(0 == --(root_element->iteration)) {
280           STACK_ELEMENT *se = root_element;
281           root_element = root_element->next;
282           free(se);
283         } else {
284           fseek(cmd_fp,root_element->position,SEEK_SET);
285           linecnt = root_element->line;
286         }
287         continue;
288       }
289       /* check if line is a setsingle statement */
290 //      if( 0 == strncasecmp(linebuffer, "setsingle ", 10)) {
291 //        if( 2 != sscanf(linebuffer+10, "%d %d\n", &address, &data) ) {
292 //          fprintf(stderr, "Syntax error on line %d.\n", linecnt);
293 //          continue;
294 //        }
295 //        init_frame(&myframe, CMD_SETSINGLE, address, data);
296 //        place_command(ser_fd, &myframe);
297 //        continue;
298 //      }
299
300       /* check if line is a clearsingle statement */
301 //      if( 0 == strncasecmp(linebuffer, "clearsingle ", 12)) {
302 //        if( 2 != sscanf(linebuffer+12, "%d %d\n", &address, &data) ) {
303 //          fprintf(stderr, "Syntax error on line %d.\n", linecnt);
304 //          continue;
305 //        }
306 //        init_frame(&myframe, CMD_DELSINGLE, address, data);
307 //        place_command(ser_fd, &myframe);
308 //        continue;
309 //      }
310
311       /* if line is none of the statements above, it's probably a syntax error */
312       fprintf(stderr, "Syntax␣error␣on␣line␣%d.\n", linecnt);
313     }
314
315     /* restore serial port settings */
316     tcsetattr(ser_fd, TCSANOW, &old_settings);
317
318     /* close serial port */
319     close(ser_fd);
320
321     /* close commandfile */
322     fclose(cmd_fp);
323
324     return EXIT_SUCCESS;
325   }
```

C.8. TimestampMatcher.c

```
1   /*****************************************************************
2    * Filter to extract measurements that are close to given
3    * timestamps
4    *
5    * USAGE: ./TimestampMatcher -f timestampfile < infile.txt > outfile.txt
6    *
7    *
8    * File:    TimestampMatcher.c
9    * Version: 0.1
10   * Date:    2014-10-09
11   * Author:  Don Mithila Palliyaguruge
12   *
13   *****************************************************************/
14
15  #include <stdio.h>
16  #include <unistd.h>
17  #include <stdlib.h>
18
19  #define BUFFERSIZE 32
20  #define LINESIZE    2
21
22  void usage(char *progname) {
23    fprintf(stderr, "\tUSAGE:\t␣%s␣-f␣timestampfile\n"
24                    "\n\tOPTIONS:"
25                        "␣-f\tfile␣containing␣the␣timestamps␣to␣match\n",
26            progname
```

```
27    );
28  }
29
30  int main( int argc, char *argv[] )
31  {
32    char          mandatoryOptions = 1;
33    FILE          *fp;
34    char          *tmstmpfile;
35    int           opt;
36    char          strTime[BUFFERSIZE];  /* ##########.###\0 */
37    char          line1[LINESIZE*BUFFERSIZE];
38    char          line2[LINESIZE*BUFFERSIZE];
39    char          *line = line1, *prevLine = line2;
40    double        imgTime, time;
41
42
43    /* evaluate commandline options */
44    while( (opt = getopt(argc, argv, "f:")) != -1 ) {
45      switch( opt ) {
46        case 'f': tmstmpfile = optarg;
47                  mandatoryOptions--;
48                  break;
49        default:  usage(argv[0]);
50                  exit(EXIT_FAILURE);
51      }
52    }
53
54    if(mandatoryOptions > 0) {
55      usage(argv[0]);
56      exit(EXIT_FAILURE);
57    }
58
59    /* open timestamp file */
60    if( (fp = fopen(tmstmpfile, "r")) == NULL ) {
61      fprintf(stderr, "Can't open timestamp file.\n");
62      return EXIT_FAILURE;
63    }
64
65    /* process the timestamp file */
66    while(NULL != fgets(line, LINESIZE*BUFFERSIZE, fp)) {
67      /* skip empty lines */
68      if(line[0] == '\n')
69        continue;
70      imgTime = atof(line);
71
72      /* process standard input */
73      while( NULL != fgets(line, LINESIZE*BUFFERSIZE, stdin)) {
74        /* skip empty lines */
75        if(line[0] == '\n')
76          continue;
77
78        /* process non-comment lines only */
79        if(line[0] != '#') {
80          /* isolate timestamp */
81          sscanf(line, "%s\t*\n", strTime);
82          time  = atof(strTime);
83
84          if( imgTime < time )
85            break;
86
87          /* backup line */
88          line     = (line == line1) ? line2 : line1;
89          prevLine = (line == line2) ? line1 : line2;
90        }
91        else {
92          /* copy comment lines */
93          fprintf(stdout, "%s", line);
94          fflush(stdout);
95        }
96      }
97      fprintf(stdout, "%s", prevLine);
98      fflush(stdout);
99    }
100   return EXIT_SUCCESS;
101 }
```

D. Battery cycle routine plan

D.1. Battery Cycle Routine Plan Without Loop

```
# Relais 1 (1)  – Zelle 1, entladen
# Relais 2 (2)  – Zelle 1, laden
# Relais 3 (4)  – Zelle 2, entladen
# Relais 4 (8)  – Zelle 2, laden
# Relais 5 (16) – Zelle 3, entladen
# Relais 6 (32) – Zelle 3, laden
# 2x 32 Zyklen 8–4–4–4, dreifache VL, SL
# Vorspann
# Alle Zellen – Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen – Laden 8 h
setport 1 42
wait 28800
# Alle Zellen – Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen – Laden 8 h
setport 1 42
wait 28800
# Alle Zellen – Ruhephase 4 h ————1
setport 1 0
wait 14400
# Alle Zellen – Laden 8 h
setport 1 42
wait 28800
# Alle Zellen – Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen – Entladung 4 h
setport 1 21
wait 14400
# Alle Zellen – Ruhephase 4 h ————2
setport 1 0
wait 14400
# Alle Zellen – Laden 8 h
setport 1 42
wait 28800
# Alle Zellen – Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen – Entladung 4 h
setport 1 21
wait 14400
# Alle Zellen – Ruhephase 4 h ————3
setport 1 0
wait 14400
# Alle Zellen – Laden 8 h
setport 1 42
wait 28800
# Alle Zellen – Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen – Entladung 4 h
setport 1 21
wait 14400
#
# ENDE
```

D.2. Battery Cycle Routine Plan With Loop

```
# Relais 1 (1)  - Zelle 1, entladen
# Relais 2 (2)  - Zelle 1, laden
# Relais 3 (4)  - Zelle 2, entladen
# Relais 4 (8)  - Zelle 2, laden
# Relais 5 (16) - Zelle 3, entladen
# Relais 6 (32) - Zelle 3, laden
# Alle Zellen - Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen - Laden 8 h
setport 1 42
wait 28800
# Alle Zellen - Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen - Laden 8 h
setport 1 42
wait 28800
#Schleife - 100 ----------------Schleife
loop 100
# Alle Zellen - Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen - Laden 8 h
setport 1 42
wait 28800
# Alle Zellen - Ruhephase 4 h
setport 1 0
wait 14400
# Alle Zellen - Entladung 4 h
setport 1 21
wait 14400
#Schleifenende ----------------Schleifenende
endloop
# Alle Zellen - Ruhephase 4 h --------SL
setport 1 0
wait 14400
# Alle Zellen - Laden 26 h
setport 1 42
wait 93600
#
# ENDE
```

D.3. Battery Charging and Discharging Cycle Plotter

```
#*****************************************************************
# * Cycle Plan visualization
# *
# * File:     Cycle2Gnuplot
#    Version: 0.1
# * Date:     2014-11-10
# * First Author:  Don Mithila M. Palliyaguruge
# *****************************************************************/

#!/bin/bash

# create a temporary directory for the named pipes
TEMPDIR=`mktemp -d`

# create the named pipes
mkfifo $TEMPDIR/in1 $TEMPDIR/in2 $TEMPDIR/timestamps $TEMPDIR/values

# duplicate stdin and pass it to the pipes (as a concurrent process)
tee < /dev/stdin $TEMPDIR/in2 > $TEMPDIR/in1 &

# transform the input
cat $TEMPDIR/in1 |dos2unix |grep -i "^wait" |sed -e "s/wait /a=a+/g" |sed -e "s/$/;a/" |bc > $TEMPDIR/timestamps &
cat $TEMPDIR/in2 |dos2unix |grep -i "^setport" |sed -re "s/setport [0-9]{1,3} /obase=4;/g" |bc |awk '{printf "%04d\n",$0}' |sed -re 's
      /[0123]?/\t&/g' |sed '{s/0/4/g};{s/1/0/g};{s/4/1/g}' > $TEMPDIR/values &
paste $TEMPDIR/timestamps $TEMPDIR/values

# remove named pipes and temporary directory
rm -rf $TEMPDIR
```

D.4. Cycle2Gnuplot.sh output file

	#Period	Cell 4	Cell 3	Cell 2	cell 1
2	14400	1	1	1	1
3	43200	1	2	2	2
4	57600	1	1	1	1
5	86400	1	2	2	2
6	100800	1	1	1	1
7	129600	1	2	2	2
8	144000	1	1	1	1
9	158400	1	0	0	0
10	172800	1	1	1	1
11	201600	1	2	2	2
12	216000	1	1	1	1
13	230400	1	0	0	0
14	244800	1	1	1	1
15	273600	1	2	2	2
16	288000	1	1	1	1
17	302400	1	0	0	0
18	316800	1	1	1	1
19	345600	1	2	2	2
20	360000	1	1	1	1
21	374400	1	0	0	0
22	388800	1	1	1	1
23	417600	1	2	2	2
24	432000	1	1	1	1
25	446400	1	0	0	0
26	460800	1	1	1	1
27	489600	1	2	2	2
28	504000	1	1	1	1
29	518400	1	0	0	0
30	532800	1	1	1	1
31	561600	1	2	2	2
32	576000	1	1	1	1
33	590400	1	0	0	0
34	604800	1	1	1	1
35	633600	1	2	2	2
36	648000	1	1	1	1
37	662400	1	0	0	0
38	676800	1	1	1	1
39	705600	1	2	2	2
40	720000	1	1	1	1
41	734400	1	0	0	0
42	748800	1	1	1	1
43	777600	1	2	2	2
44	792000	1	1	1	1
45	806400	1	0	0	0
46	820800	1	1	1	1
47	849600	1	2	2	2
48	864000	1	1	1	1
49	878400	1	0	0	0
50	892800	1	1	1	1
51	921600	1	2	2	2
52	936000	1	1	1	1
53	950400	1	0	0	0
54	964800	1	1	1	1
55	993600	1	2	2	2
56	1008000	1	1	1	1
57	1022400	1	0	0	0
58	1036800	1	1	1	1
59	1065600	1	2	2	2
60	1080000	1	1	1	1
61	1094400	1	0	0	0
62	1108800	1	1	1	1
63	1137600	1	2	2	2
64	1152000	1	1	1	1
65	1166400	1	0	0	0

E. Video Frames of Measurement Series

E.1. Measurement Series 13

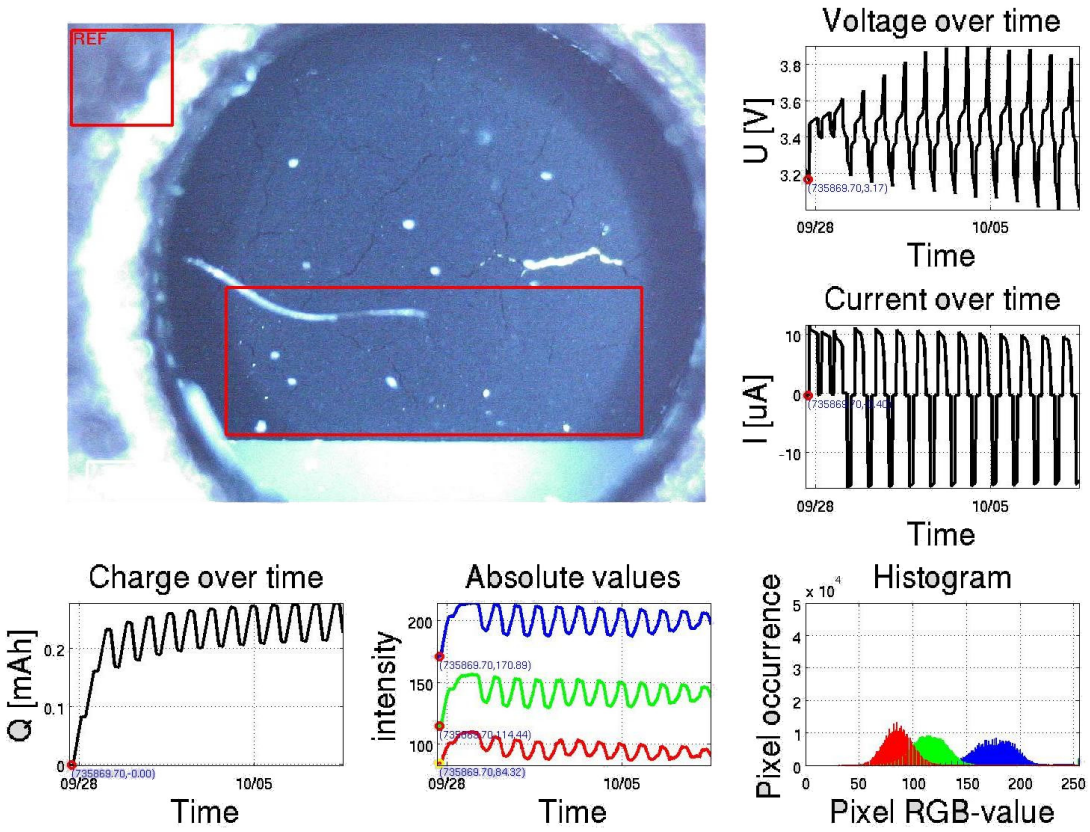


Figure E.1.: A single frame of a video designed for electrical and optical measurements of measurement series 13 cell 3. Frame contents cell voltage, current and charge as for electrical measurements. Color intensity and histogram of affected region (large rectangle in red) displayed as optical measurements.

E.2. Measurement Series 15

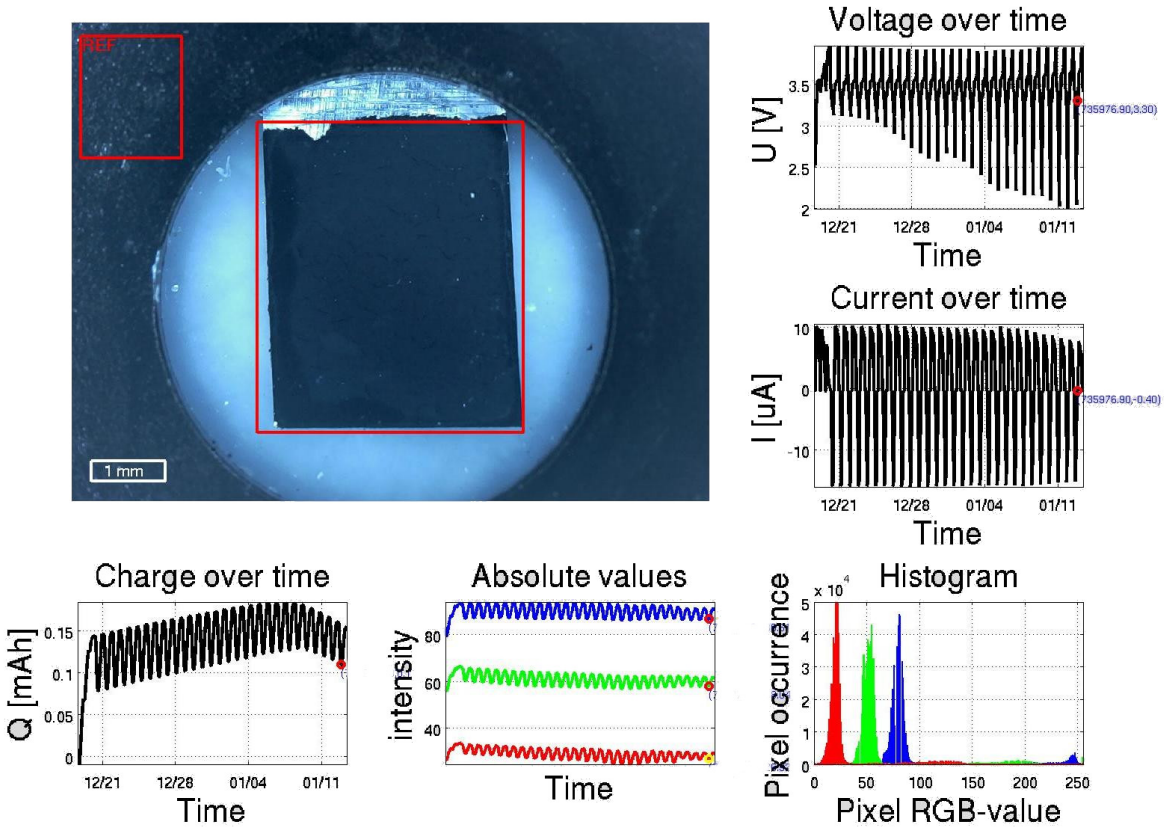


Figure E.2.: A single frame of a video designed for electrical and optical measurements of measurement series 15 cell 1. Frame contents cell voltage, current and charge as for electrical measurements. Color intensity and histogram of affected region (large rectangle in red) displayed as optical measurements.

E.3. Measurement Series 17

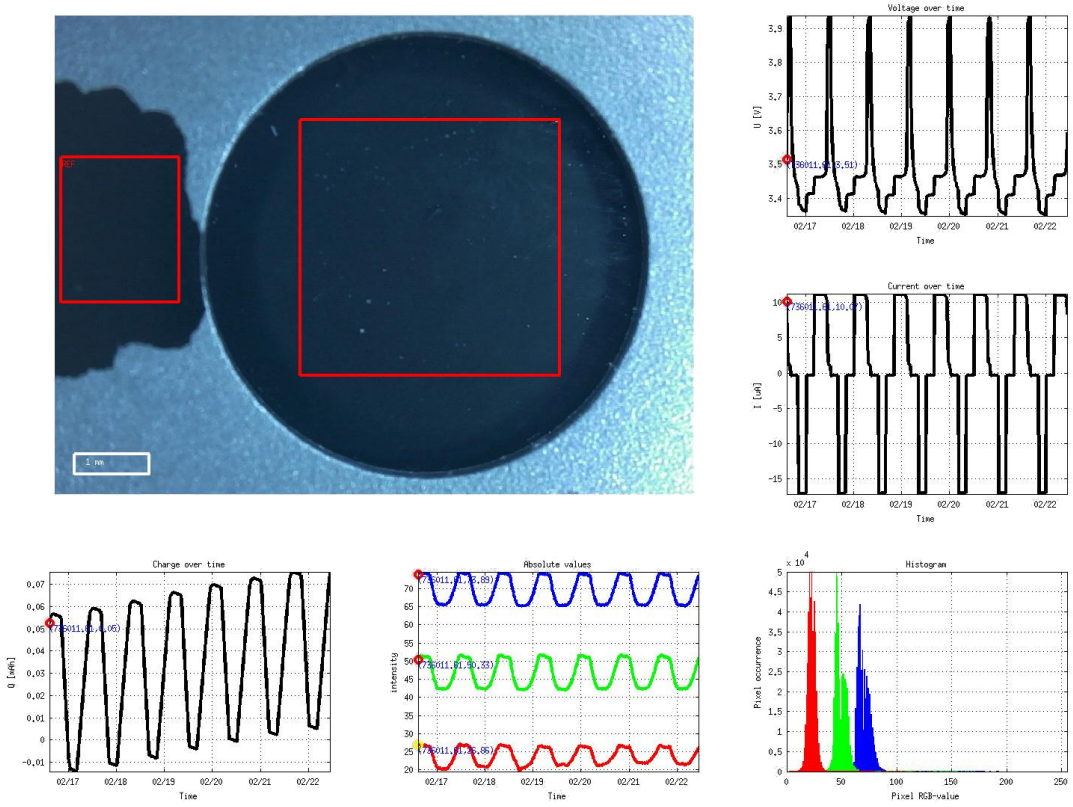


Figure E.3.: A single frame of a video designed for electrical and optical measurements of measurement series 17 cell 2. Frame contents cell voltage, current and charge as for electrical measurements. Color intensity and histogram of affected region (large rectangle in red) displayed as optical measurements.

F. Electrical Measurements Evaluation

MR	Date	Discharge(Ohm)				Charge(Ohm)			
		Cell 1	Cell 2	Cell 3	Cell 4	Cell 1	Cell 2	Cell 3	cell 4
13	27.09.2014	46979	47025	47132	n.u.	9934	9883	9944	n.u.
14	26.11.2014	46979	47025	47132	n.u.	9934	9883	9944	n.u.
15	17.12.2014	20000	20129	20125	20090	46979	47025	47011	47132
16	20.01.2015	20000	20129	20125	20090	46979	47025	47011	47132
17	12.02.2015	20000	20129	20125	20090	46979	47025	47011	47132

Table F.1.: Charge and discharge - resistor values (n.u. - not used).

F.1. Measurement Series 13

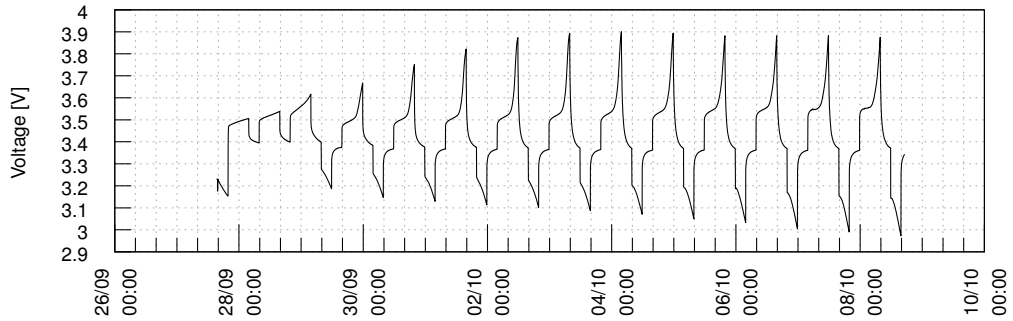


Figure F.1.: Measurement Series 13 - Voltage in Volt

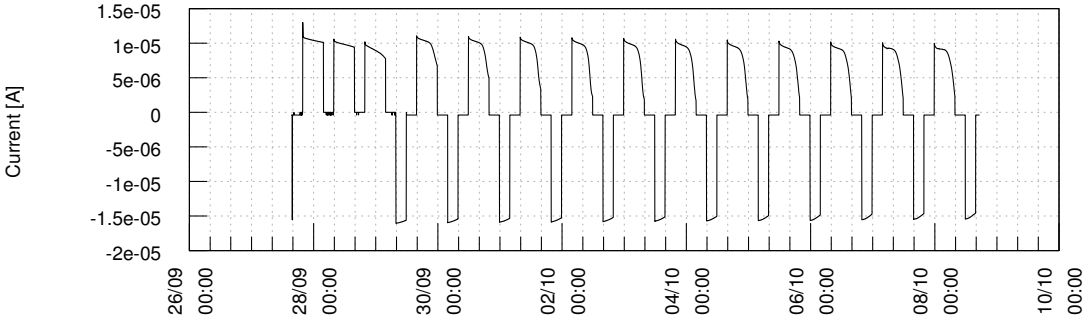


Figure F.2.: Measurement Series 13 - Current in Ampere

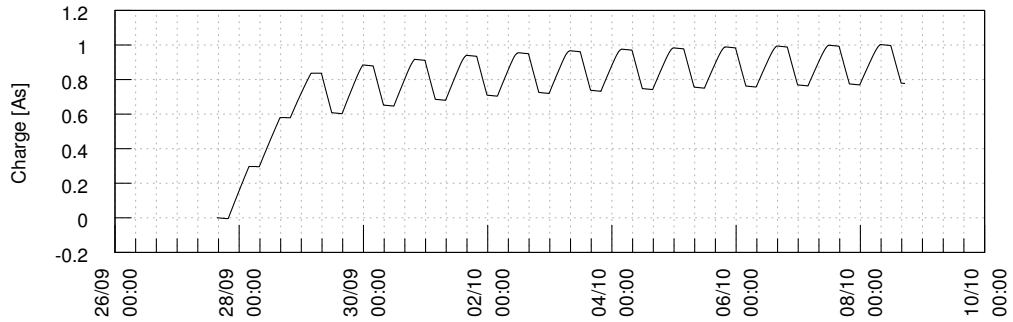


Figure F.3.: Measurement Series 13 - Charge in Ampere seconds. During charge, rest and discharge periods, there is a permanent discharge of the cell due to the voltmeter's interior resistance of 10 MΩ resulting in a discharge current of 300 nA.

F.2. Measurement Series 14

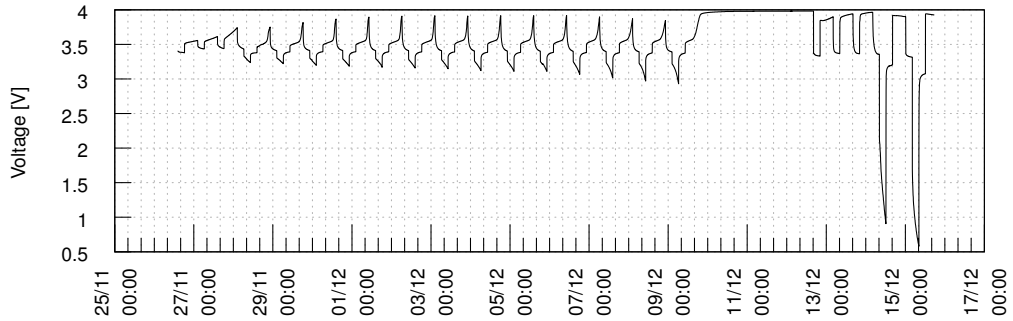


Figure F.4.: Measurement Series 14 - Voltage in Volt

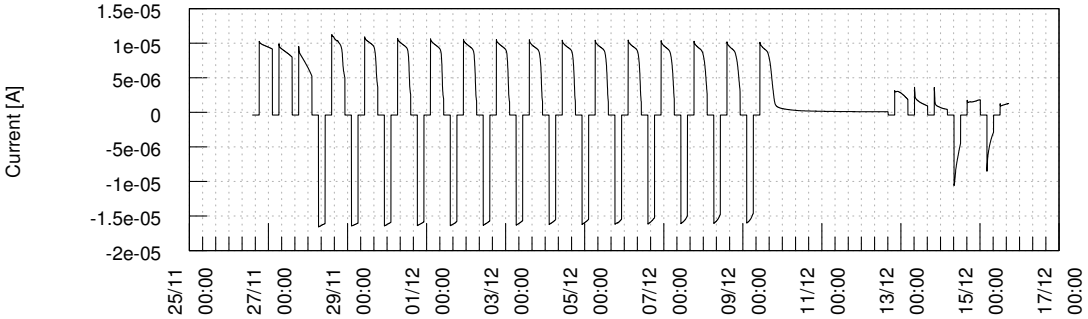


Figure F.5.: Measurement Series 14 - Current in Ampere

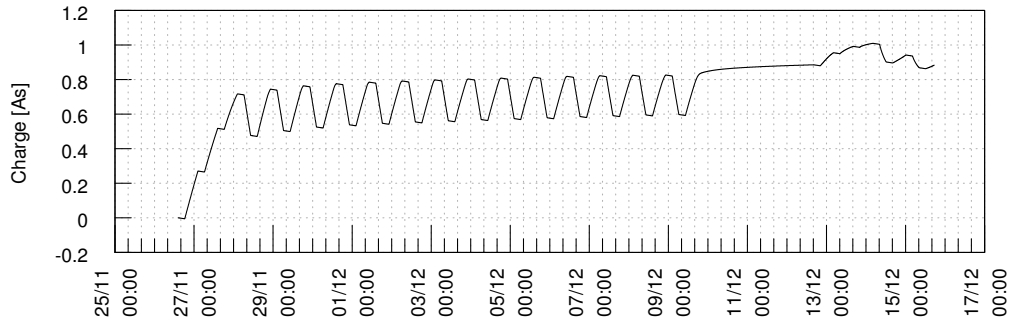


Figure F.6.: Measurement Series 14 - Charge in Ampere seconds. During charge, rest and discharge periods, there is a permanent discharge of the cell due to the voltmeter's interior resistance of 10 MΩ resulting in a discharge current of 300 nA.

F.3. Measurement Series 15

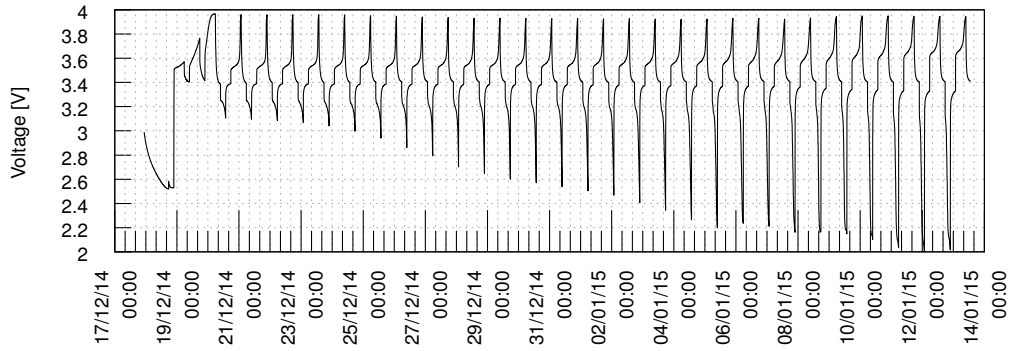


Figure F.7.: Measurement Series 15 - Voltage in Volt

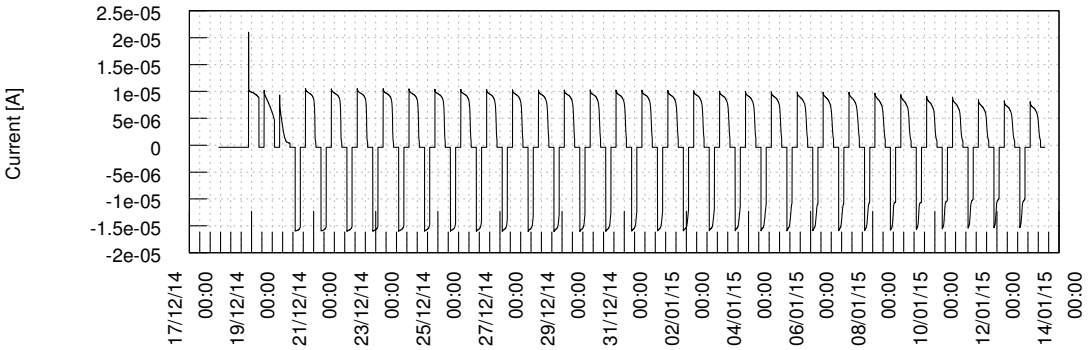


Figure F.8.: Measurement Series 15 - Current in Ampere

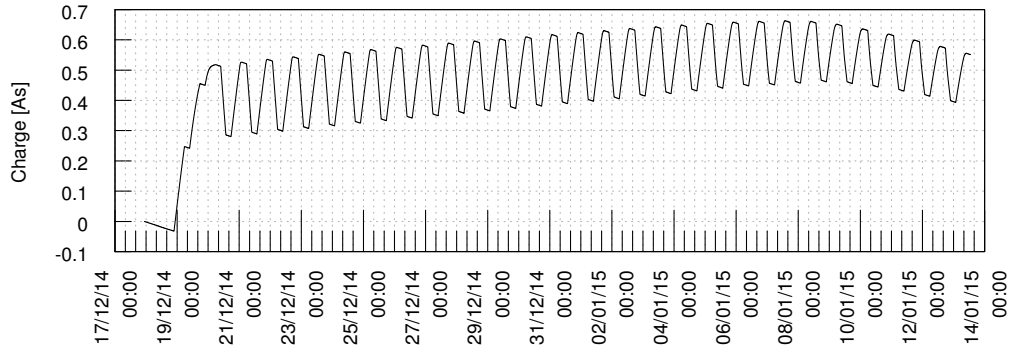


Figure F.9.: Measurement Series 15 - Charge in Ampere seconds. During charge, rest and discharge periods, there is a permanent discharge of the cell due to the voltmeter's interior resistance of 10 MΩ resulting in a discharge current of 300 nA.

Declaration

I declare within the meaning of section 25(4) of the Ex-amination and Study Regulations of the International De-gree Course Information Engineering that: this Bachelor report has been completed by myself inde-pendently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Hamburg, March 6, 2015

City, Date

sign