Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorthesis

Ali Emami

## Investigation of optical effects in lithium batteries with image processing and the optical flow

*Fakultät Technik und Informatik*
*Department Informations- und Elektrotechnik*

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

Ali Emami

# Investigation of optical effects in lithium batteries with image processing and the optical flow

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. Karl-Ragmar Riemschneider
Second Examiner : Prof. Dr.Ing. Stefan Lehmann

Day of delivery 20. April 2017

**Ali Emami**

## Title of the Bachelorthesis

Investigation of optical effects in lithium batteries with image processing and the optical flow

## Keywords

Lithium iron phosphate battery, Electrical and optical measurements, Edge detection, Denoising, Greyscaling, Binarization, SAP removal, Optical flow, Horn Schunck method, Middlebury optical flow benchmark

## Abstract

Direct observation of the chemical and physical battery state by visualization of the spatial and temporal distribution of the effect during charging and discharging process of the battery test cells. This visualization is done by first doing the image preprocessing techniques and then implementing the optical flow methods such as Horn and Schunck method on battery test cell images. It also produces videos together with significant diagrams for further scientific mass data analysis.

**Ali Emami**

## Titel der Arbeit

Untersuchung der optischen Effekte in Lithium-Ionen-Batterien mittels Bildverarbeitung und Optical-Flow-Analyse

## Stichworte

Lithiumeisenphosphat-Batterie, Elektrische und optische Messung, Kantenerkennung, Rauschreduktion, Grauwertbildung, Binarisierung, S&P Entfernung, Optischer Fluss, Horn-Schunck-Methode, Middlebury Benchmark

## Kurzzusammenfassung

Diese Arbeit beschreibt die direkte Beobachtung des chemischen und physikalischen Batteriezustands durch Darstellung der örtlichen und zeitlichen Verteilung des Effekts während des Lade-und Entladevorgangs der Batterietestzelle. Diese Darstellung wird zunächst erreicht durch Bildverarbeitungtechnik und danach durch die Anwendung der Optischen Flow-Methode nach Horn-Schunck, angewendet auf Bildsequenzen der Batterietestzelle. Es werden Videosequenzen zusammen mit aussagefähigen Diagrammen für weitere wissenschaftliche Analysen von Massendaten gewonnen.

# Acknowledgment

I would like to take this opportunity to express my appreciation to Prof Dr.Karl-Ragmar Riemschneider and Prof.Dr.Stefan Lehmann for the continuous support, patience, motivation, and immense knowledge during my research and study at HAW Hamburg.

My special thanks to Mr.Dipl.-Phys.Valentin Roscher. His guidance helped me in all time of research and writing of this thesis.

I would also like to thank Mr.Müller for his support during the correction of this bachelor thesis.

Last but not least, with all my heart would like to thank Babaji who taught me the value of hardwork and an education. Without him, I may never have gotten to where I am today. I know he is up there, listening, watching over me and sending me his blessings constantly and is my guardian angel.

# Contents

# List of Tables

# List of Figures

# 1. Introduction

## 1.1. Motivation

In electric vehicles, batteries with many cells are used to supply the power. The battery is controlled by a battery management system which needs measurement data from each individual cell. Common battery monitoring concepts use only electrical measurements and estimate the battery state of charge based on a calculated battery model. These models based on electrical parameters have significant 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 charging.
- Long term changes of cell parameters due to aging.

## 1.2. Project

Due to the mentioned reasons, the battery research group BATSEN at HAW Hamburg does research for direct observation of the chemical and physical battery state. In previous work an optical effect was discovered, which is promising for that purpose. For lithium iron phosphate cathodes the change of reflection intensity is observed and reproduced. The observations were done with microscope cameras and specialized test cells. A large amount of image data is available from previous experiments done by Jan Grießbach, Philipp Schiepel and others. The first try of image processing and preprocessing for investigation of effects in Lithium battery electrodes were conducted by Don Mithila Meshan Palliyaguruge. The final goal of the BATSEN group is to introduce a complete optical and electronic sensor system in the battery cells by using LEDs and optical fibres to observe and evaluate optical effects of the battery cells and exchange the data with the battery management system [18].

## 1.3. Assignment

The main objective of the thesis is to visualize the spatial and temporal distribution of the effect during the charge and discharge processes. In particular, the method of optical flow should be tested first. It is done by developing Matlab program modules for image processing, data evaluation and visualization of these image data. The platform is a conventional PC with Linux.

Chapter "Theory", first gives an introduction to the lithium iron batteries and the idea of state of charge with the measurement setup and their corresponding electrical and optical results. Then it describes the theory behind the image preprocessing techniques together with examples. At last the theory of optical flow methods with their mathematical calculations and relevant examples are explained.

Chapter "Preprocessing of images", implements the preprocessing techniques on the battery test cell images. The results are used for the optical flow implementations in the next chapter.

Chapter "Optical flow", implements the different optical flow methods such as Horn and Schunck, Lucas Kanade and Middlebury first on the test images with and without background and then on the battery test cell images. The comparisons of implementations are discussed in mass data analysis chapter.

Chapter "Mass data analysis", contains the analysis and information of the battery test cells of four different data sets together with their final optical and electrical data after implementation of the preprocessing techniques and optical flow methods on the test cell images. At the end it presents the snapshots of the created videos which show the changes of electrical and optical parameters over the set of test cell images during charging and discharging process. The created video is copied to the CD which is attached to this bachelor thesis. At the end the section of scientific analysis will compare the implemented optical flow methods.

Chapter "Conclusion", will conclude the thesis by first explaining the assumptions and introducing the discussions together with the summary of the thesis. At the end there is a guidance for the further development of the assignment.

# 2. Theory

## 2.1. Lithium iron battery

Lithium ion batteries is the most popular energy storage system for a vast variety of portable electronic devices including laptops, smart phones with the world market valued at ten billion dollars. Recently, they became one of the top candidate for large scale power source for electric vehicles since they have high energy density, no memory effect, long cycle life, and low self discharging. Lithium iron battery consists of some parts called cells. Like all batteries these cells which are made out of electrochemical cells converts chemical energy into potential electrical current. In order to flow the energy (electron) from the battery to the electrical circuit every ion will move between negative (anode) and positive (cathode) terminals of each cell in a electrolyte chamber. Lithium iron battery technology is aimed for the secondary battery type which are rechargeable electrochemical batteries which can be used for portable electronics equipments and also electric vehicles [11] [18].

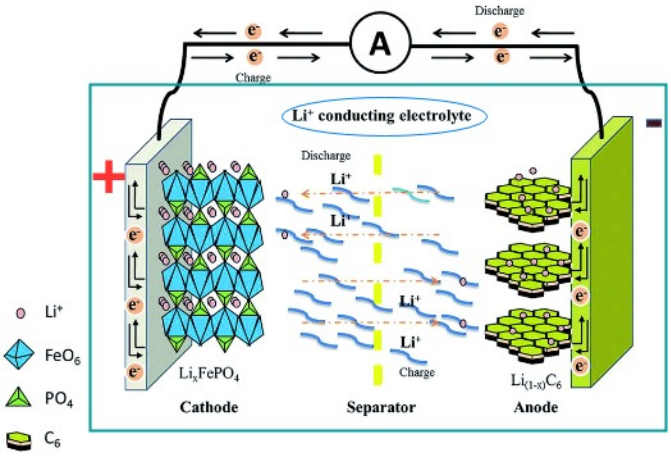



Figure 2.1.: During the charging process the Li+ ion moves to the negative electrode and electrons moves out as well. The separator avoids the contact of electrodes and will enable ionic transport. Original image from [11] is modified.

Referring to the figure above, the electrical current will move towards the battery cell by the conductive surfaces. The positive electrode (cathode) is pure lithium metal oxide and the negative electrode (anode)is graphite. The battery cell is filled with electrolyte which enable ionic transport. By charging process of the battery cell, Li+ ions will flow from the cathode through the non aqueous electrolyte, carrying the current towards the graphite structure of the anode. By discharging process since the energy is removed from the cell, lithium ion will migrate back to the cathode and the battery will be discharged.

Chemical equation of the positive electrode [13]:

$$LiFePO_4 \overset{Charging}{\rightleftharpoons} FePO_4 + Li^+ + e^- \tag{2.1}$$

Chemical equation of the negative electrode [13]:

$$Li^+ + e^- \overset{Charging}{\rightleftharpoons} Li \tag{2.2}$$

Comprehensive chemical equation [13]:

$$LiFePO_4 \overset{Charging}{\rightleftharpoons} FePO_4 + Li \tag{2.3}$$

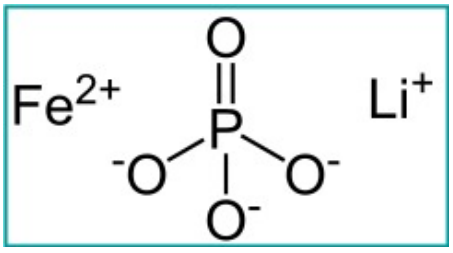



Figure 2.2.: Chemistry structure of lithium iron phosphate. Original image from [17] is modified.

The first picture shows the photo of a rechargeable lithium iron phosphate battery which is purchased from ECC batteries and is used in the BATSEN research group at HAW Hamburg. The second picture shows the photo of the layers inside the shown battery. As it is shown in figure 2.1 , it consists of cathode (lithium iron phosphate), anode (graphite) and the separator. The aluminum current collector and copper current collector work as conductive surfaces in cathode and anode respectively. The separator is made of micro porous polymer membrane which is permeable for small Li+ ions and will prevent electrical short circuit [18].

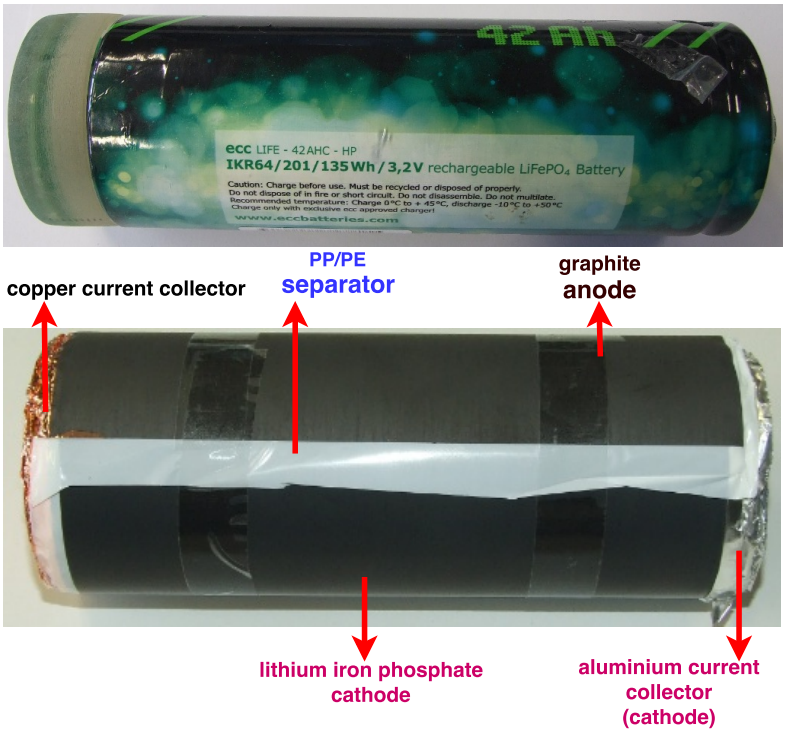



Figure 2.3.: Casing and the internal architecture of a commercial lithium iron phosphate battery. Left side is negative contact, right side is positive contact. **Top:** Rechargeable lithium iron phosphate battery, product of ECC batteries. **Bottom:** Layers inside the battery cell.

## 2.2. Battery state of charge

Rechargeable lithium iron battery needs an efficient battery management system in order to have a stable operating time and to avoid possible dangerous cases. One of the challenges of the battery management systems is detection of the chemical state of the battery. Determining the state of charge (SoC) of the battery can be done by the electrical measurement of voltage and current which are fast and accurate, but have to rely on current integration and voltage measurements to predict the state of charge and the intrinsic problems such as integration errors can violate this prediction.

During the charging and discharging process of the lithium battery, chemical reaction will happen on electrodes and electrolyte composition. Direct optical observation during the charging process of the lithium battery shows a measurable changes on the graphite anode and lithium iron phosphate as the metal oxide for the cathode. see the figure below:

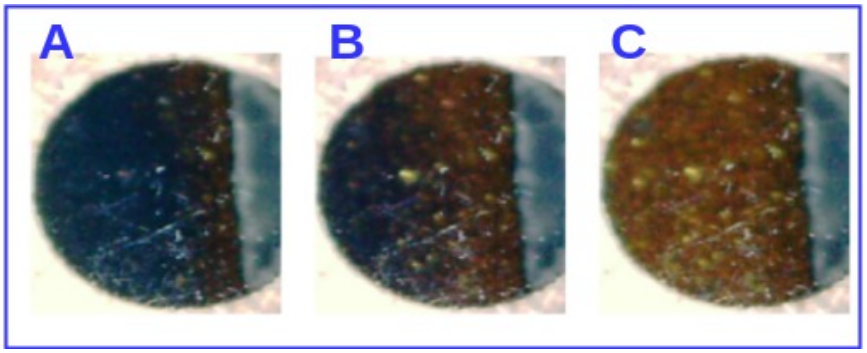



Figure 2.4.: **A** : Beginning of the effect, **B** : After 98 minutes, **C** : After 373 minutes. These series show the discoloration of the anode by the charging process of the Li-ion battery cell which is result of intercalation of Li+ ion into graphite [18].

By considering these captured changes in the graphite layered structure, BATSEN research group at HAW Hamburg determined cathode state by optical measurement of the electrodes which is independent of the electrical measurement parameters. The next section will introduce the measurement steps.

## 2.3. Measurement setup

This chapter explains the three different measurement steps which are performed in order to gain the electrical and optical data that can be used for visualization of the battery state of charge in relation to the optical effects and electrical parameters and implementing the optical flow methodology.

Measurement setups are divided into three steps:

- Test cell production
- Electrical data recording
- Optical data recording

**Test cell production** introduces the methods and steps with the measurement tools and adjustments which are used for the production of the lithium iron phosphate battery cells in the laboratory. The produced test cells will be used for further optical and electrical observations and measurements during this thesis.

**Electrical data recording** introduces the electrical schematic and measurement tools and methods which were used to measure the voltage, current and charge of the battery test cells during the charging and discharging process.

**Optical data recording** introduces the method and tools which were used to capture the images of the battery test cells during the charging and discharging process. Then presents the measurement plots for voltage, current and charge over the period of time.

All of the electrical and optical measurements were done to observe the relation of the electrical parameters and optical effects on lithium ion battery during charging and discharging. In this experiment measurement setups are based on the modeled lithium ion battery environment by using ECC-Opto-Std test cells.

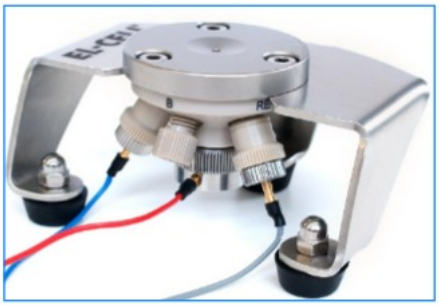


Figure 2.5.: ECC-Opto-Std Optical test cell. Original image from [19] is modified. ECC-Opto-Std test cell is a model cell made by the company EL-Cell GmbH which the battery can be built inside and the optical behavior of the electrode materials during charging and discharging of the battery can be captured by a camera. Figure 2.4 depicts the ECC-Opto-Std and figure 2.5 shows its internal architecture [18].

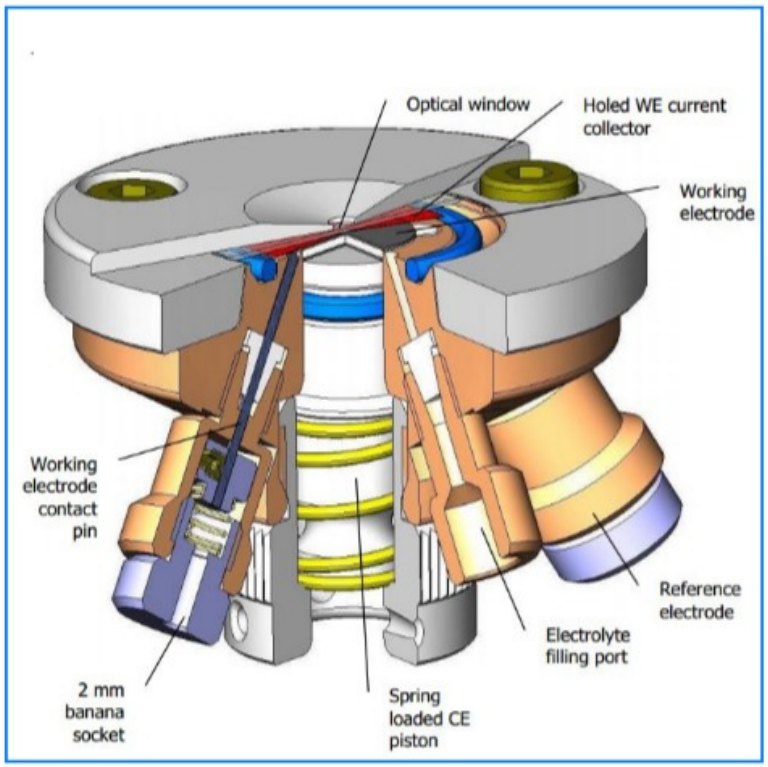



Figure 2.6.: Internal architecture of ECC-Opto-Std. Original image from [19] is modified.

### 2.3.1. Test cell production

Samples for testing is created from lithium ion phosphate (LFP) powder, carbon powder, and indium tin oxide (ITO) powder is used to produce the test cell and place it in the ECC-Opo-Std for optical observation during charging and discharging process. Several possible placements of the electrode samples within the test cells are shown in the figure below:

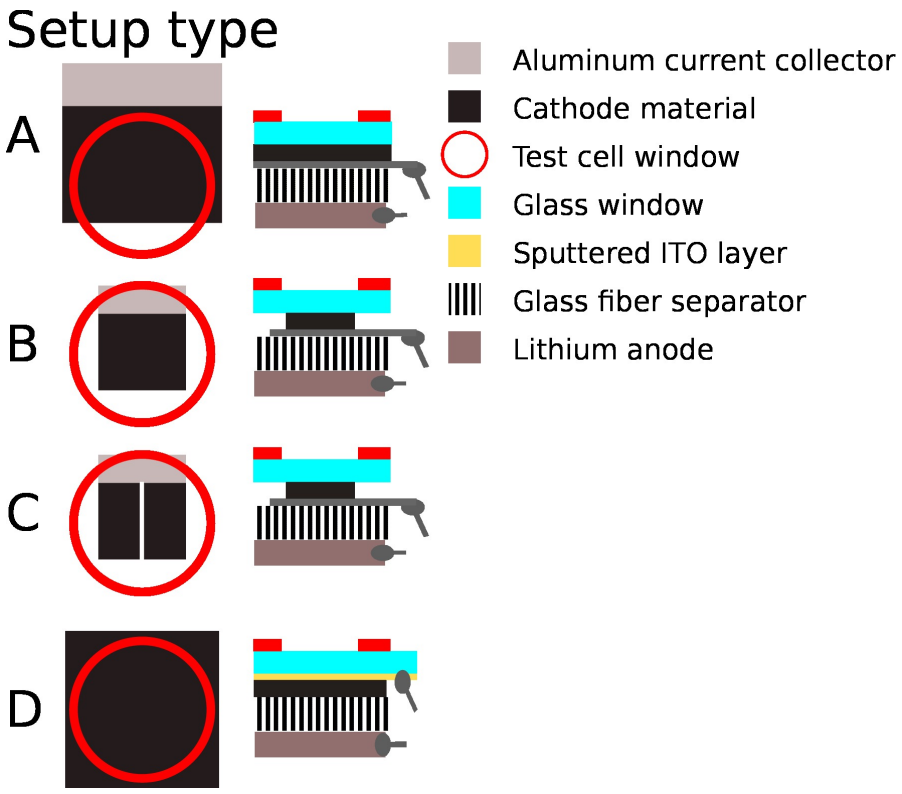



Figure 2.7.: Placements of the electrodes within the test cell. The camera monitors the area within the red circles. Setup types **A** includes one electrode, which is just partially clear through the test cell lid. Setup type **B** is much the same, but permits the monitoring of the complete area. In setup **C**, two electrode sheets with different properties are placed side-by-side of each other. They can be electrically connected, but will only share the counter electrode. In setup types **A-C**, separator is the white area neighboring the electrodes. In setup type **D**, the minimum current collector is recouped by a transparent layer of ITO. Therefore, the backside electrode can also be visible. In this case the separator is not visible [22].

### 2.3.2. Electrical measurement setup

To measure voltage and current during the charging and discharging of the battery test cell a test bench is created with the following circuit for one cell as shown below :

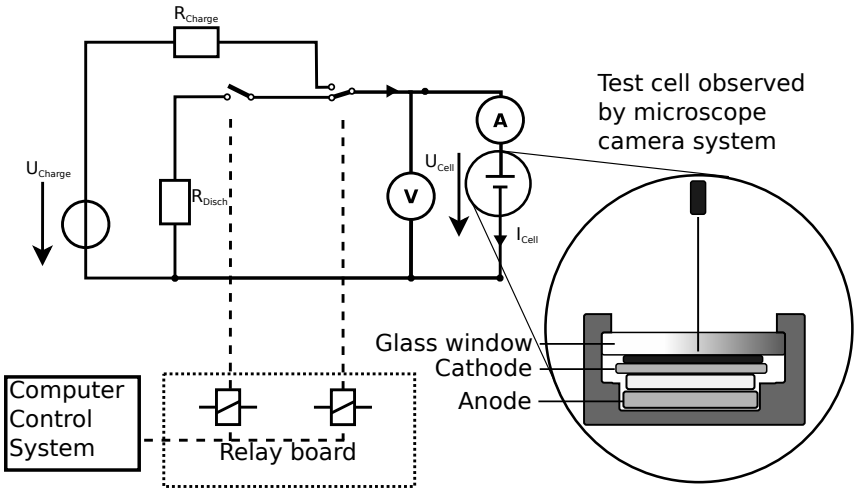



Figure 2.8.: Schematic circuit for measuring voltage and current of one test cell. This is the first version in which switches are controlled by a relay board (PCB) (modified from [21]).

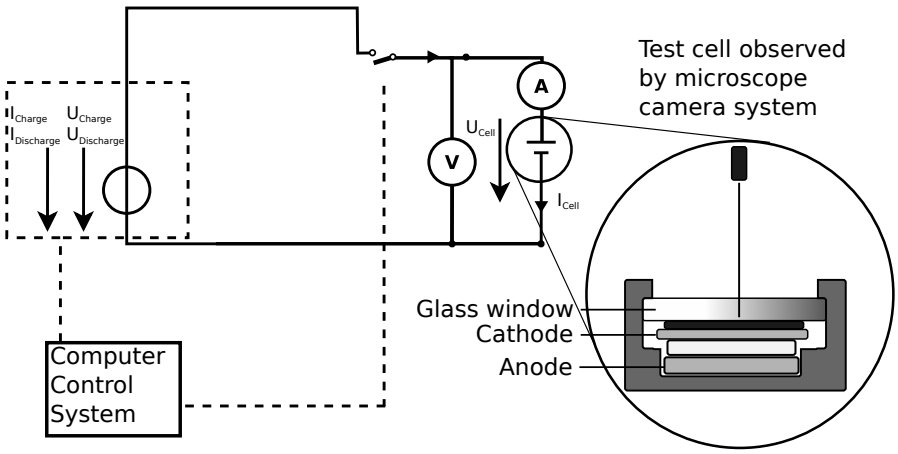



Figure 2.9.: Schematic circuit for measuring voltage and current of one test cell. This is the second version in which voltage and current are controlled individually by a control system and a custom control and measurement board (modified from [21]).

Test cells are continuously charged and discharged by using a measurement platform on a single board computer. The voltage and current of the cell are measured and therefore the charge of the cell can be calculated. The reflectance calculated from the digital images, averaged over a region of interest from the observed electrode surface. Figure below presents the data from a full measurement run [18] [22].

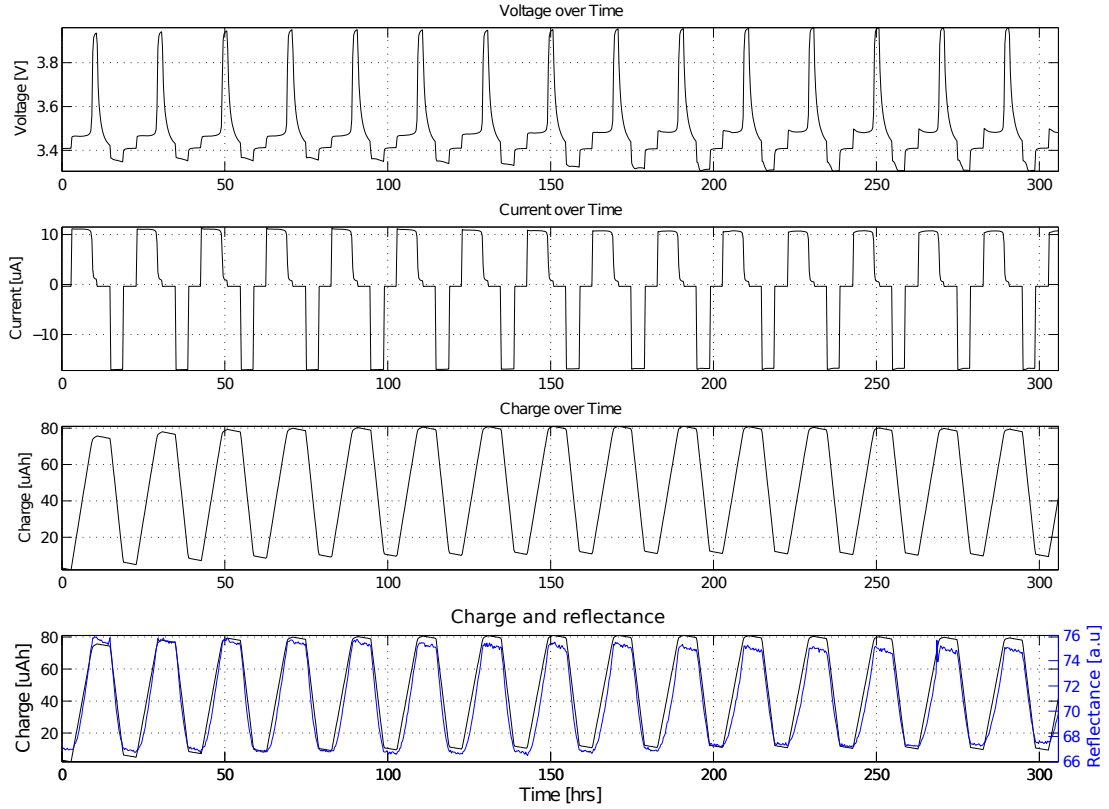



Figure 2.10.: **Top and center**: Test cell voltage and current for setup type D. **Bottom**: Reflectance (blue line) and charge (black line)of the same cell, it shows that they are proportional and in phase. This correlation proves that a sensing method for direct charge detecting is applicable [22].

In all measurements on sample test cells including marker-enhanced LFP, almost a good linear correlation between the state of charge and the measured optical effect is detected (figure below) [22] :

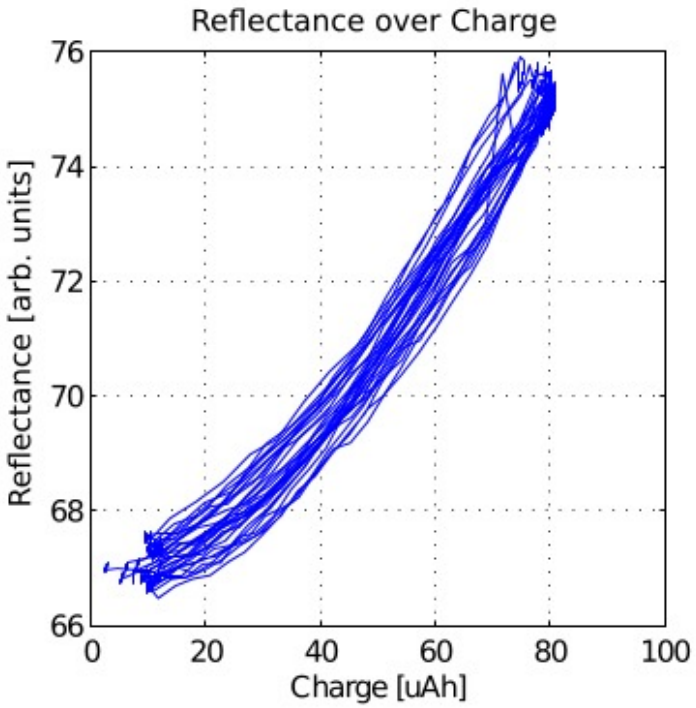



Figure 2.11.: This graph shows the optical reflectance vs cell charge of many cell cycles shown in figure 2.10. There is relatively a linear dependency between reflectance and charge. With the help of this correlation and optical observation on battery electrode cell state can be determined [22].

### 2.3.3. Optical measurement setup

Images of the test cells are taken by optical microscope camera every 10 seconds and save them in a lossless color format. Some image processing techniques are used to reduce the effect of aging of the illumination LED. Image stacking is performed to minimize the influence of noise.

To assure constancy of the results, measurements were operated over the course of hours, days per weeks. In the images, regions of interest which are normally includes the full visible electrode area are defined manually . Figure below shows the test bench including the camera at the top of the ECC-Opto-Std which contains the lithium iron battery test cell [22].

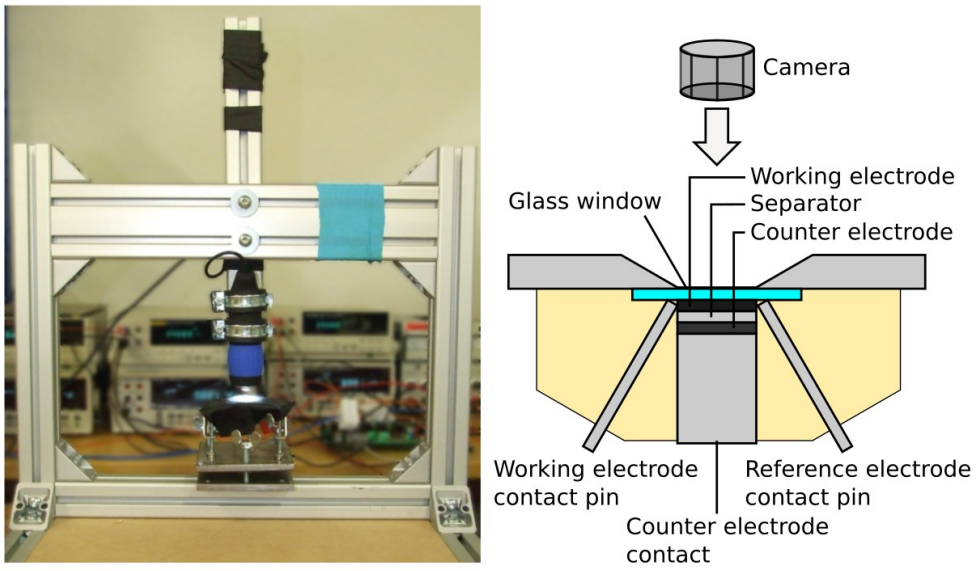



Figure 2.12.: **Left side:** Measurement setup with the microscope camera at the top and the test cell with glass window. The camera will capture photos during charging and discharging of the test cell.
**Right side:** Windowed test cell with the camera at the top [22].

Photos below show the optical results on different setup types made on figure 2.7 :

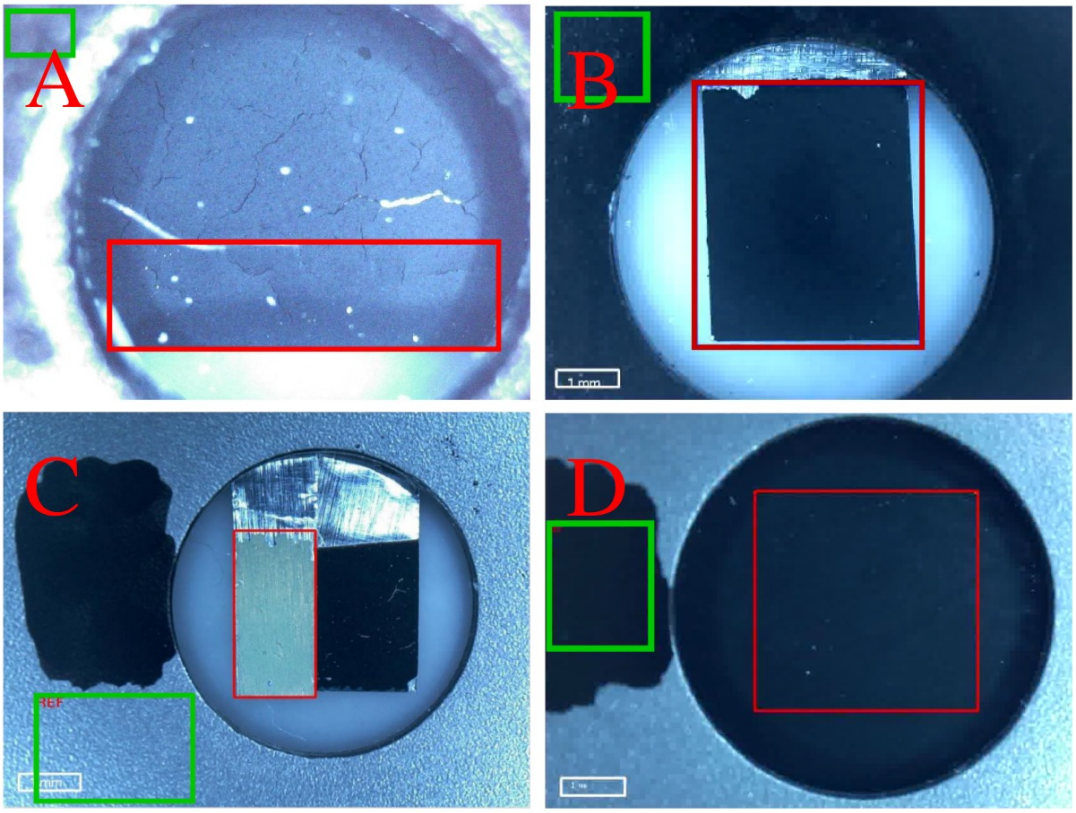



Figure 2.13.: Optical results of the test cells made in four different setup types A-D as described in figure 2.7. The images are blue due to the usage of the LED lighting during capturing the images.
Red rectangles: Evaluated electrode sections which are the area of interest to observe the optical changes during the charging and discharging process and later is used for optical flow implementations.
Green rectangles: Reference sections needed to correct the changes in illumination and also used as reference area to compare with the area which a lot of optical changes happens [22]. Optical result of the test cell made in setup type C is not used for further image preprocessing and optical flow implementations.

Below shows the captured images of a test cell during discharging process, a visible change in reflectance can be observed.

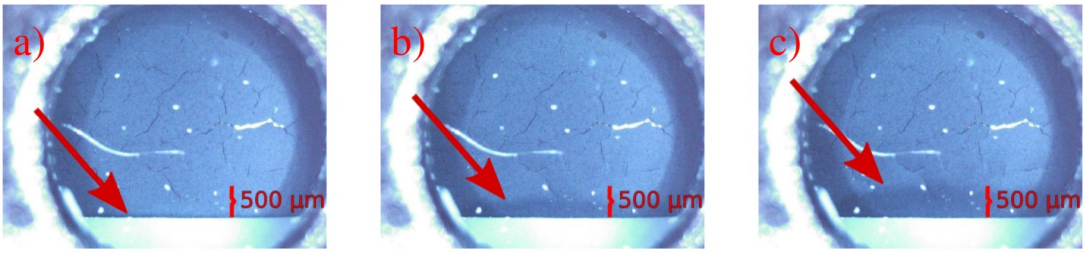



Figure 2.14.: Reflectance change in the test cell images of setup type A (see figure 2.7) during the discharging process of electrode.
**Top part:** The electrode
**Bottom part(white section):** The separator

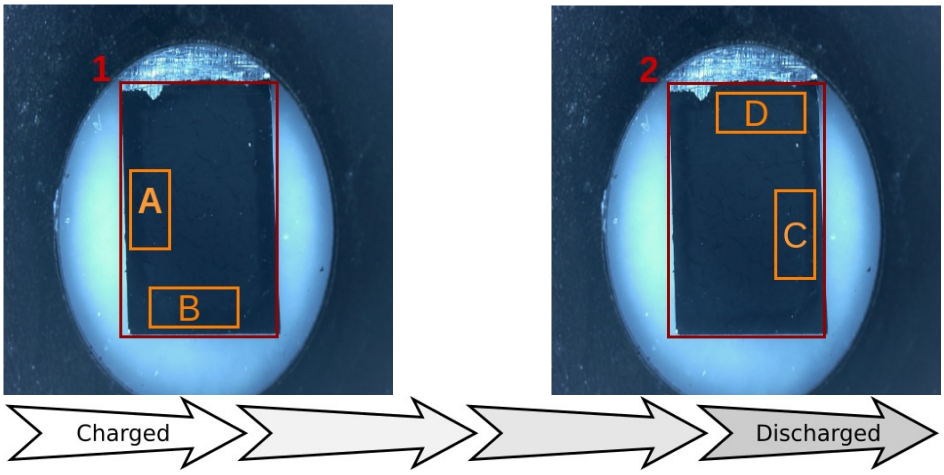



Figure 2.15.: Profiles of test cell images (the two red rectangles shows the area of interest with labeled sections **A**,**B**,**C**,**D** candidates area for the optical flow implementations. At these selected areas the major optical changes are observed.

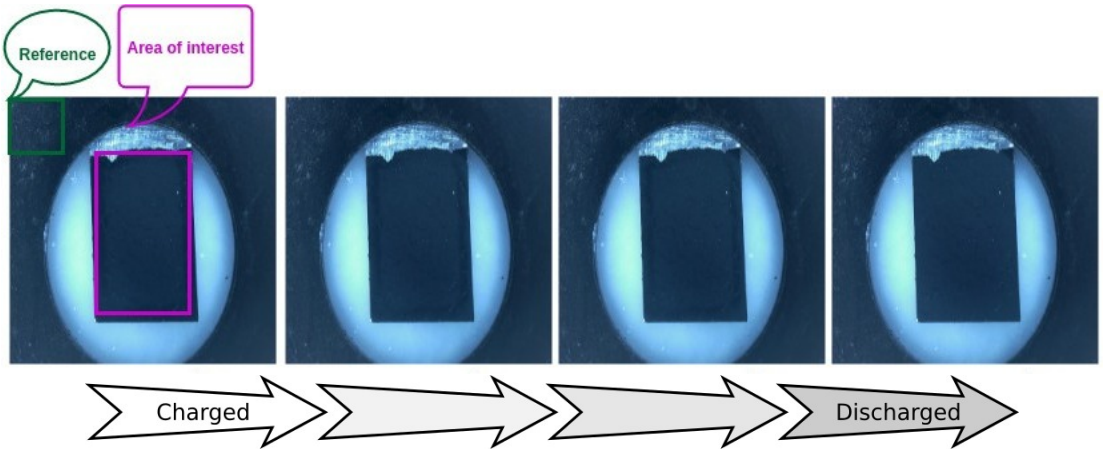



Figure 2.16.: Full images of a battery test cell during discharging process.

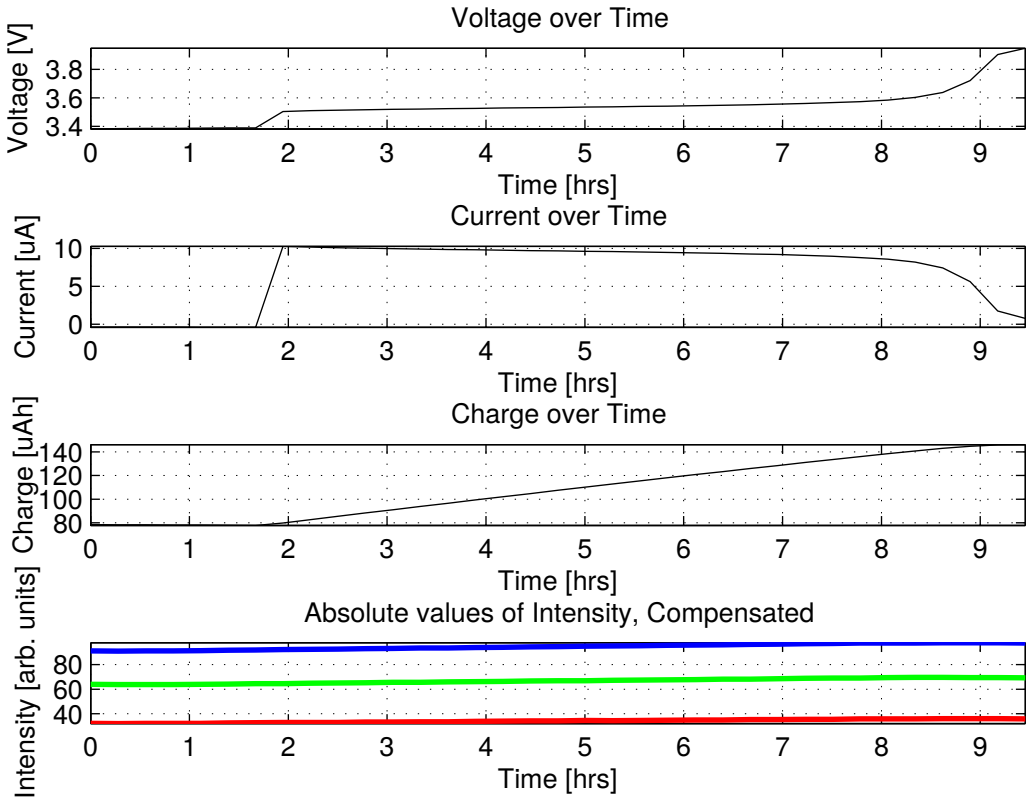



Figure 2.17.: Voltage, current and intensity changes over the period of time during the charging process.

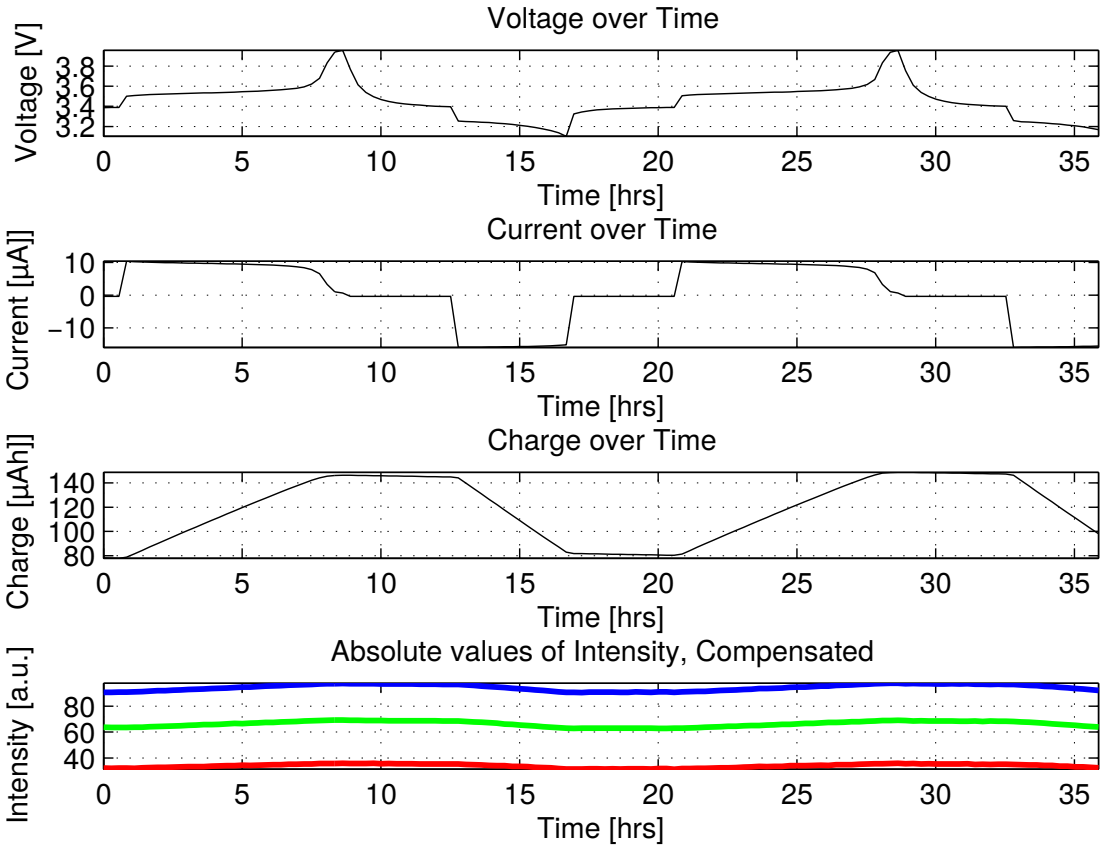



Figure 2.18.: This figure shows the voltage, current and intensity changes over the period of time during the complete charging and discharging process. The color intensity of the image changes as well and this optical change is divided into three main color channels (RGB), the reason that the intensity of the blue color is higher compare to green and red is basically due to the usage of the LED lighting during capturing the images.

## 2.4. Preprocessing of images

The purpose of the preprocessing of the images is the improvement of the image data that suppresses unwanted distortions or enhance some image features which are important for further processing, in this case implementing the optical flow methods [25].

This chapter introduces the theory behind the preprocessing methods which are used on the battery test cell images. The implementation part presents the result of the preprocessing techniques.

### 2.4.1. Edge detection

Edge detection is a branch of segmentation theory which detects sudden discontinuities by converting the two dimensional images into set of curves. It is used as one of the preprocessing steps of the battery cell images to capture the sharp intensity(brightness)changes during charging and discharging of the battery.

An edge detector algorithm receive discrete, digitized battery cell images as the input and produces the so called edge map as output to be able to observe the intensity changes and sharpnesses on the battery cell images. The edge map presents explicit information about the position and strength of the edges.

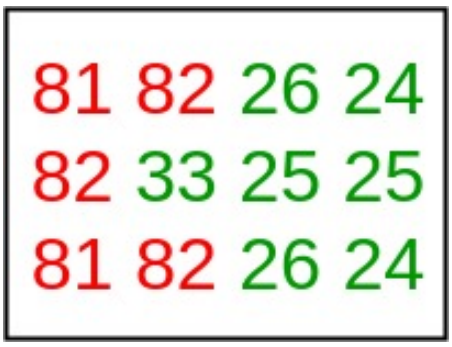



Figure 2.19.: Edge detection by looking for a neighborhood with lots of changes. Each pixel in an image is represented by matrix value, which shows the intensity of each pixel. By comparing the neighboring pixel values which has high differences the edges (sharp intensity changes) in the image can be detected.

**Necessary criteria for a good edge detection algorithm are the following:**

- Edge detection algorithm should detect as many edges as possible in the battery cell images with low error rate.
- The operator that detect the edge point has to precisely localize on the center of the edge.
- The battery cell images include noises, these noises should not create false edges. Each edge has to be marked only once in the edge detection algorithm [15].

**Steps in edge detection:**

- Compute the local derivatives.
- Magnitude of the first derivative can be used to detect the presence of an edge.
- The sign of the second derivative can be used to determine whether an edge pixel lies on the dark or light side of an image.
- Zero crossing of the second derivative is at the midpoint of a transition in gray level, which provides a powerful approach for locating the edges.
- Mark points where the gradient magnitude is large with respect to neighboring points. Ideally this yields curves of edge points.

Considering the fact that derivatives are sensitive to noise therefore, it is required to smooth images before determining image gradients by using a derivative of Gaussian filter.

One of the most reliable and simple edge detection methods is canny edge detection algorithm because of its good detection and localization with clear response.

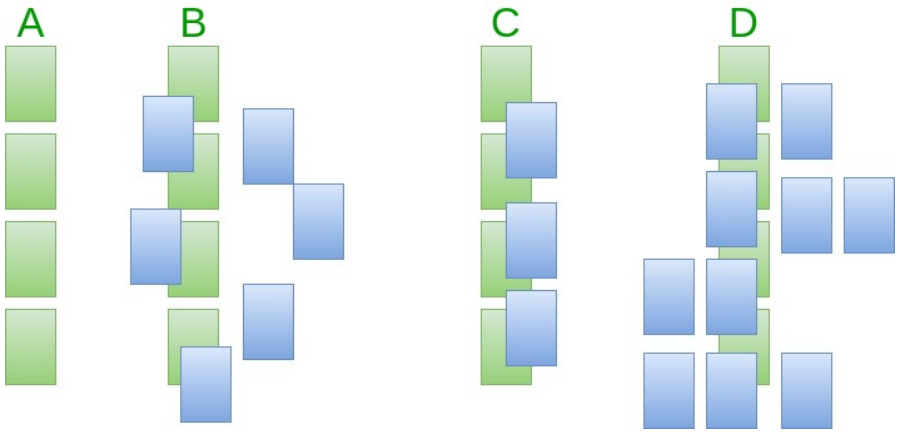



Figure 2.20.: A: True Edge, B: Poor robustness to noise, C: Poor localization, D: Too many response

Canny edge detection use the calculus of variations which is a technique to find the function that optimizes a given functional. This edge detector operator uses a multi-stage algorithm to detect a wide range of edges in the test cell images. The optimal function in Canny's detector can be obtained by the first derivative of a Gaussian.

**Steps of canny edge detection algorithm [15]:**

- Use Gaussian filter to smooth the test cell images and remove the possible noises.
- Calculate the intensity gradient of the test cell image.
- Use non maximum suppression to avoid spurious response to edge detection.
- Apply double threshold to find potential edges.
- Track the edge by hysteresis method: suppress all the other edges that are weak and not connected to strong edges.

Parameters of canny edge detection algorithm which can adjust the calculation time and the efficiency of the algorithm [8]:

- Size of the Gaussian filter: result of the canny algorithm depends on the smoothing filter which is used in the first step. The smaller the filter, the less blurring results with detectable small and sharp lines are available.
- Thresholds: using two threshold with hysteresis offer more flexibility than using only single threshold. Setting the threshold is a critical point because by setting it too high, it can lose some information and by setting it too low, it will find irrelevant

information and noises. The threshold has to set base on the constraints of the battery test cell images.

### 2.4.2. Denoising

Battery test cell images are taken with digital cameras which will pick up noise from a variety of sources. Further use of these images for optical flow implementations and analysis of the results will require that the noise be (partially) removed. Denoising introduces a lot of new methods for removing Gaussian random noise. The main aim of these methods is to reduce the noise level, while preserving the image features and edges as much as possible [6].

Theory of denoising introduces wide variety of methods for removing Gaussian random noise. The necessary criteria for the chosen method is to reduce the noise level, while preserving the features of test cell images such as edges as much as possible.

One of the techniques which is often used in image processing to remove the noise and recover the blurred images is by using generalized inverse filtering. However, this technique is very sensitive to additive noise. In order to be able to remove the additive noise and invert the blurring simultaneously the wiener filtering is introduced which is an optimal trade off between inverse filtering and noise smoothing [5].

In theory the wiener filter is one of the classical linear filtering which works by reducing the mean square error between the calculated random process and the desired process. Wiener filter consists of two main parts, an inverse filtering part and a noise smoothing part. It does the deconvolution by inverse filtering and also discard the noise by a compression operation [1]. This theory is implemented as one of the preprocessing steps on the battery test cells images by using the wiener filter function available in Mathematica software. Removing Gaussian noise from image by applying a range-r Wiener filter using a (2r+1)x(2r+1) convolution kernel [12]: **WienerFilter[image,r]**.

### 2.4.3. Grayscaling

Grayscaling method is used as one of the preprocessing steps of the battery test cell images. The reason for using this method is that after gray scaling the RGB images, the value of each pixel is a single sample and less information needs to be provided for each pixel of the image. Therefore, there is no need to use more complicated and harder-to-process RGB images. The grayscaling Matlab code and result is included in the next chapter [14].

### 2.4.4. Binarization

Binarization or thresholding is one of the image segmentation methods and in this thesis is used as a preprocessing step on the test cell images before implementing the optical flow methodologies.

The theory is that if a pixel value of an image is bigger or smaller than a defined threshold value, then it can be only be converted to two possible values(zero or one). This binary value is presented by black and white color in the result image. Its algorithm converts a pixel image to a binary image by first converting the image into grayscale and then applying the threshold. Image binarization is most effective in the images with high contrast level [7][24].

The binarization algorithm and the Matlab code and the result of its implementation on the battery test cell images are all added and explained in the next chapter.

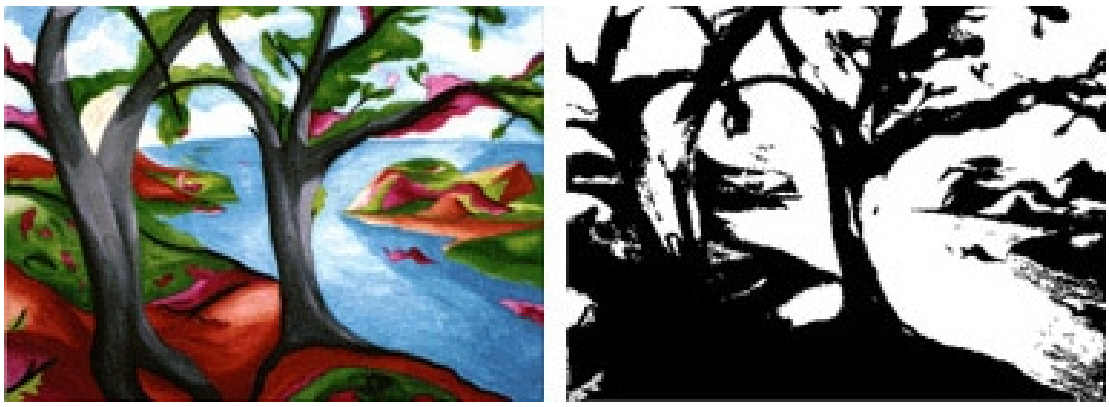


Figure 2.21.: Left side shows the original image and the right side the corresponding binarized image [16].

## 2.5. Optical Flow

Optical flow reflects the changes due to motion at a certain pixel position from first two dimensional image to second during the period of time. Its velocity field represents the magnitude and direction of change of the object points across a two dimensional image. Consider the image below [23]:

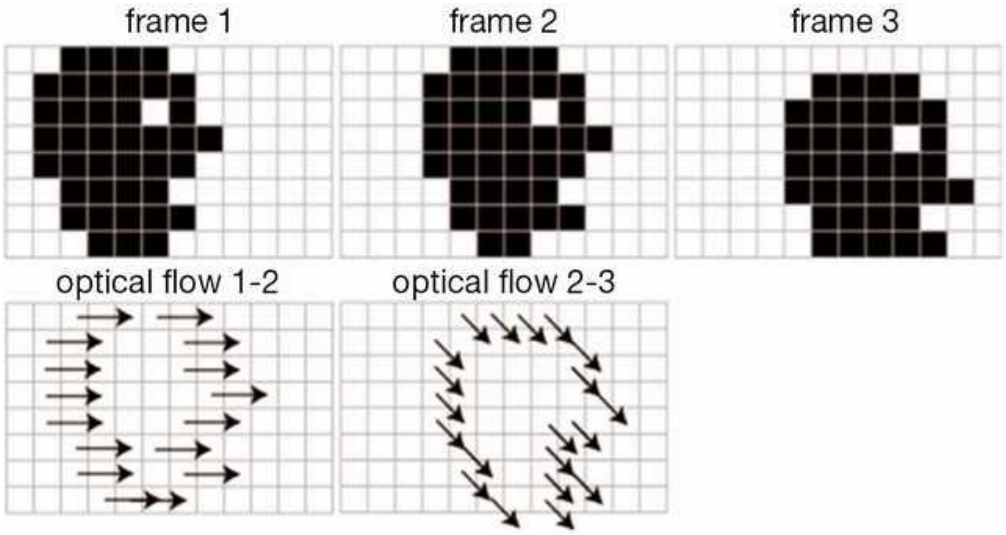



Figure 2.22.: By sampling the structured light spatially and temporally it will result in an image sequence. This figure shows three frames, which represents the movement of the silhouette of a head. The optical flow is shown as the correspondence of contour pixels between frame 1 and frame 2 as well as frame 2 and frame 3. For optical flow estimation methods, the challenge is to find out the point which correspond each pixel in the image, not only the contour pixels [20].

**Assumptions to consider in optical flow theory [2] :**

- The pixel intensities of an object don't change during consecutive frames.
- Pixels that are close to each other have very similar motion.

The optical flow method estimate the motion between two images that are taken at times t and $t + \Delta t$ at every pixel position. These methods comes from the local Taylor series approximations of the image signal and use partial derivatives according to the spatial and temporal coordinates [9].

In a 2D+t dimensional form a pixel at location $(x, y, t)$ with intensity $I(x, y, t)$ will move by $\Delta x$, $\Delta y$ and $\Delta t$ between the two images, and "brightness constancy constraint" can be given as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

By considering small movement, the image constraint at $I(x, y, t)$ with Taylor series can be calculated to get:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \mathrm{higherorderterms}$$

Continue these equations it shows that:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0$$

$$or$$

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0$$

Then gives in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

where $V_x, V_y$ are the $x$ and $y$ components of the velocity or optical flow of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the image derivatives at $(x, y, t)$ in the given directions. $I_x$,$I_y$ and $I_t$ can be written for the derivatives in the following.

Thus:

$$I_x V_x + I_y V_y = -I_t$$

$$or$$

$$\nabla I^T \cdot \vec{V} = -I_t \tag{2.4}$$

This equation has two unknowns and cannot be solved. This issue is addressed as the aperture problem of the optical flow algorithms. In order to calculate the optical flow another set of equations are needed, with some additional constraint. Methods of optical flow require additional conditions for calculating the actual flow [3]. See also Eq. 2.4.

**Methods for determining optical flow:**

- Horn Schunck method
- Lucas Kanade method
- Middleburry

### 2.5.1. Horn Schunck method

The Horn Schunck (HS) method is used to find optical flow by estimating the direction and speed of a moving object from one image to another. The algorithm assumes smoothness in the flow over the whole image. Therefore, it reduces noise in flow [9].

The flow is formulated and the following function is given for two dimensional image sequences :

$$E = \int\int \left[(I_x u + I_y v + I_t)^2 + \alpha^2(\| \nabla u \|^2 + \| \nabla v \|^2)\right] \mathrm{d}x\mathrm{d}y$$

$I_x$, $I_y$ are the derivatives of the image intensity values along the x, y and and $I_t$ is time dimensions, $\vec{V} = [u(x, y), v(x, y)]^\top$ is the optical flow vector, and $\alpha$ is a regularization constant. Increasing the value of $\alpha$ makes the flow smoother. This functional can be reduced by the associated multi-dimensional Euler-Lagrange equations. These are:

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} = 0$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x}\frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial v_y} = 0$$

Where $L$ is the integrand of the energy expression, which leads to:

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0$$

$$I_y(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0$$

Where subscripts shows partial differentiation and $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ shows the Laplace operator. The laplacian is estimated by using finite differences, and can be written as $\Delta u(x, y) = \overline{u}(x, y) - u(x, y)$ where $\overline{u}(x, y)$ is a weighted average of $u$ calculated

in a neighborhood around the pixel at location (x,y). By using this notation the above equation can be written as:

$$(I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \overline{u} - I_x I_t$$

$$I_x I_y u + (I_y^2 + \alpha^2)v = \alpha^2 \overline{v} - I_y I_t$$

Which is linear in $u$ and $v$ and can be calculated for every pixel in the image. Every time the neighboring values of the flow field is changed, it has to be updated. Therefore, the iterative scheme is derived as the following:

$$u^{k+1} = \overline{v}^k - \frac{I_x(I_x\overline{u}^k + I_y\overline{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v^{k+1} = \overline{v}^k - \frac{I_y(I_x\overline{u}^k + I_y\overline{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

Where $k+1$ denotes the next iteration, which has to be calculated and $k$ is the previous result.

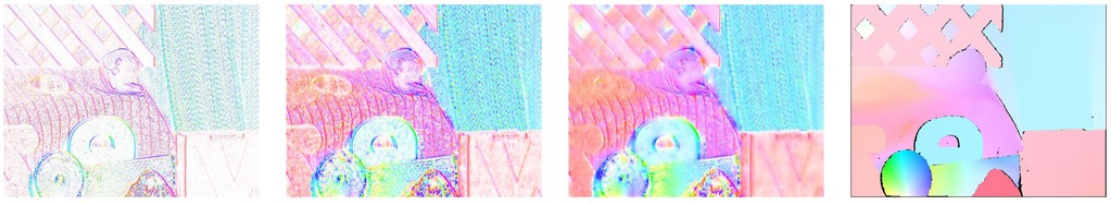



Figure 2.23.: Comparison of the results depending on the number of algorithm iterations [10].

Horn Schunck algorithm returns a high density of flow vectors, i.e. the flow information missing in inner parts of homogeneous objects is filled in from the motion boundaries. on the other side, it is more sensitive to noise than local methods [9].

### 2.5.2. Lucas Kanade method

The optical flow calculation performed by Lucas and Kanade which is a differential method and is based on the constancy of image brightness. It assumes that for a motion (u,v) of a point in an image I the brightness of the point does not change:

$$I(x, y, t) = I(x + u, y + v, t + 1) \tag{2.5}$$

By using the first order Taylor expansion it results to the following gradient constraint equation:

$$I_x u + I_y v + I_t = 0 \tag{2.6}$$

This uncertainty is resolved by a least squares estimate over an image given by :

$$(uv)^T = [\sum (I_x I_y)^T (I_x I_y)]^{-1} \sum I_t (I_x I_y)^T \tag{2.7}$$

Since It is also less sensitive to image noise than point-wise methods. and also, since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image[4].

### 2.5.3. Middlebury optical flow benchmark

Middlebury test bench mark is a method of the optical flow implementations on two consecutive images. In this case two battery test cell images during the charging or discharging process. The purpose of the Middlebury optical flow benchmark is to visualize the intensity changes at one point between two consecutive images. As a result, it produces the vector plot and color map which present the direction of optical changes through the images.

Middlebury code is developed by Department of Computer Science, Brown university and is acknowledged as a benchmark system for optical flow algorithms. In the basic implementations, it allows the use of the Horn Schunck method as well as an optimized method called "Classic-NL" that uses improvements such as image preprocessing to improve on the Horn Schunck method [26].

"Classical" is an algorithm which is basically derived from the original Horn Schunck formulation, and by using different techniques the model and method can differ [27].

Implementation of this method together with examples are explained and shown in the optical flow chapter.

# 3. Preprocessing of images

In this chapter the implementation of the preprocessing methods and feature extraction techniques on the test cell images are shown and explained with the figures and images. These preprocessing steps are needed first to visualize the optical changes in the test cell images and the result is used for further implementations of the optical flow methods.

Figure 3.1 shows the original test cell images during the charging process before implementing the preprocessing techniques. The corresponding measured electrical parameter and intensity of the RGB data of the image is also added.


Figure 3.1.: This figure shows the steps of the battery test cell image processing. Initially the original image is preprocessed with the explained techniques and the result will be used for the optical flow implementations and at the end as a result, the vector plots represents the optical changes from one test cell image to another. Presentation of the corresponding color map of the vector plot is also possible.

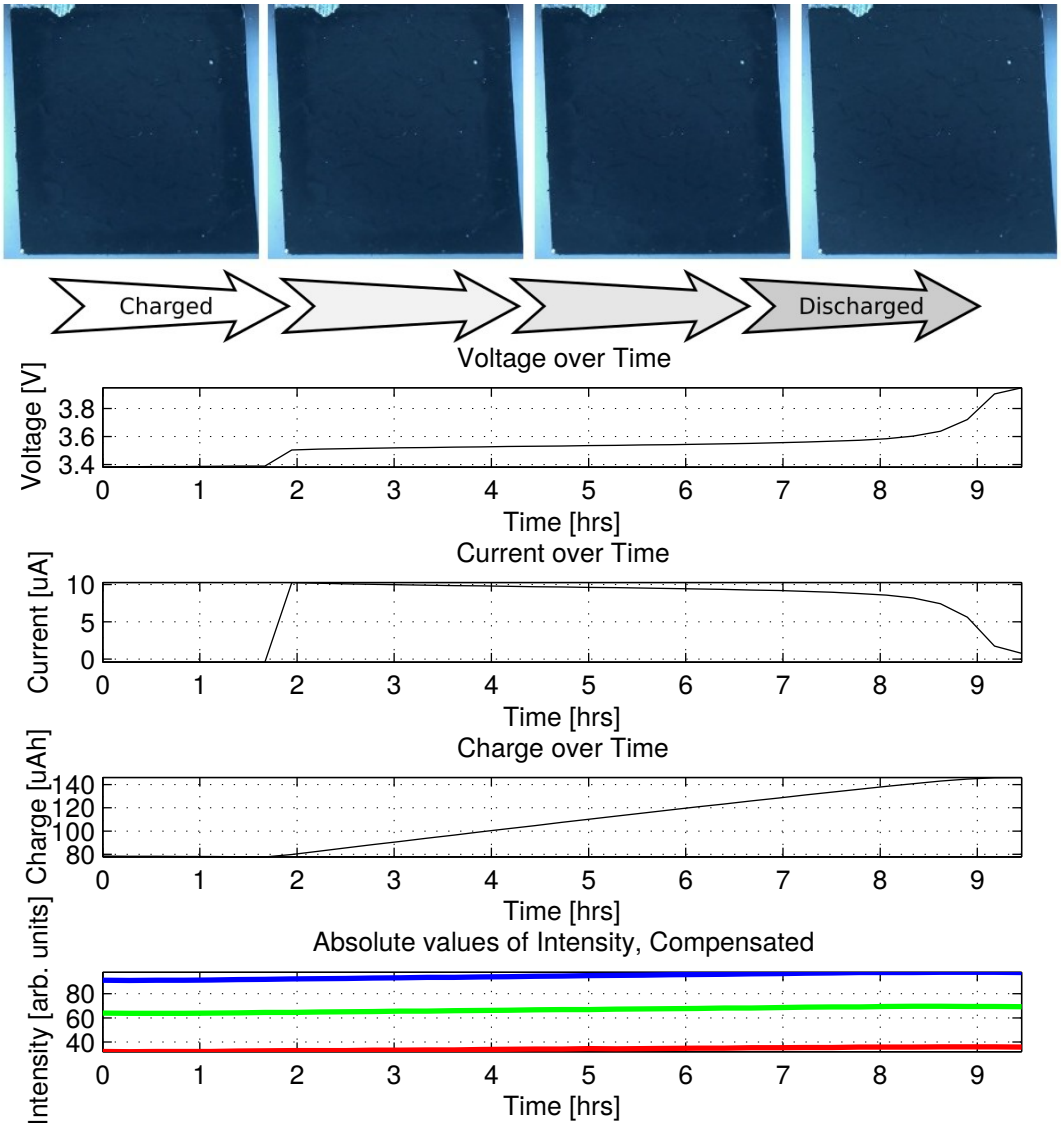



Figure 3.2.: The first row from the top shows the original test cell images which only focus on the area of interest (where the optical changes happens). The bottom part shows the corresponding electrical parameters of the test cell and the color intensity of the images during the charging process.

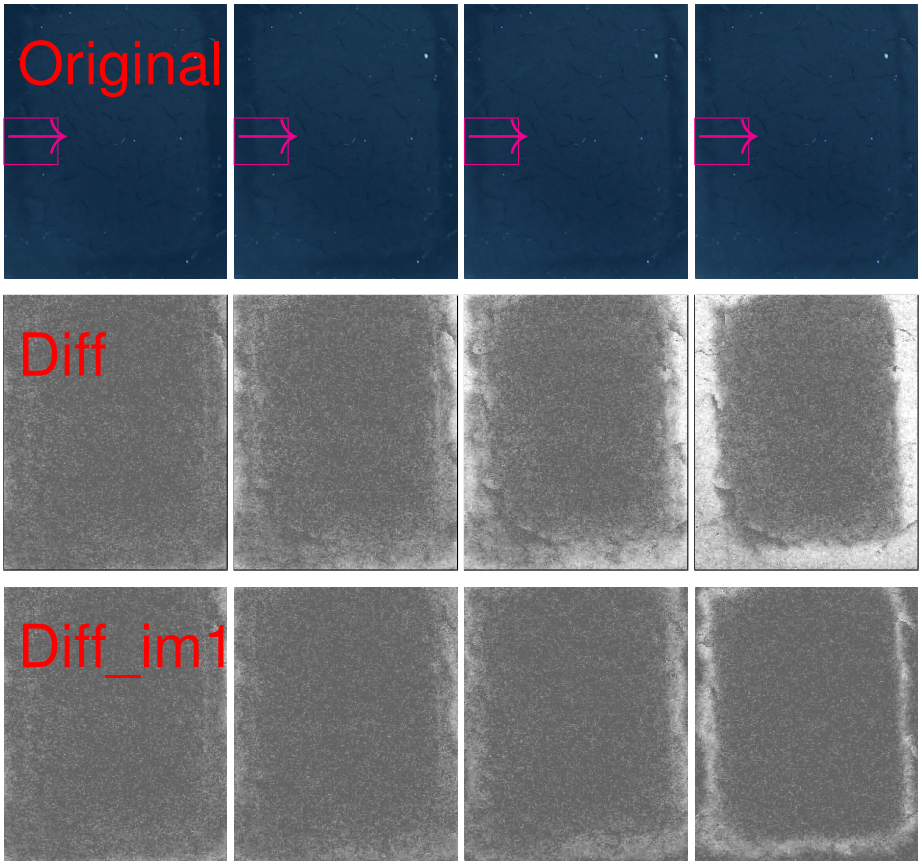



Figure 3.3.: Preprocessing of the electrode image taken during the charging process. **Leftmost image:** Completely discharged, **Rightmost image:** Completely charged. **Original**: Raw images without performing image processing. **Diff**: Calculating the overall change from the raw data by subtracting a static image from every image. **Diff_im1**: Calculating the temporal gradient from the new data by subtracting the previous image from every image. This shows that in the beginning intercalation primarily appears close to the electrode interface, and then during the measurement intercalates further into the electrode. Pink boxes in **Original** indicate area evaluated for profiles. A cell of setup type B is used for the measurements [22].

## 3.1. Greyscaling and Noise reduction

In order to implement the grey scaling method as the preprocessing step of the test cell images, the *rgb2grey* Matlab function is used. In this case, the grey scaled test cell image is further processed by reducing the noise with the help of a "remove salt and pepper filter". The corresponding Matlab filter is called *medfilt2*. Figure 3.6 presents the result of the implementations.

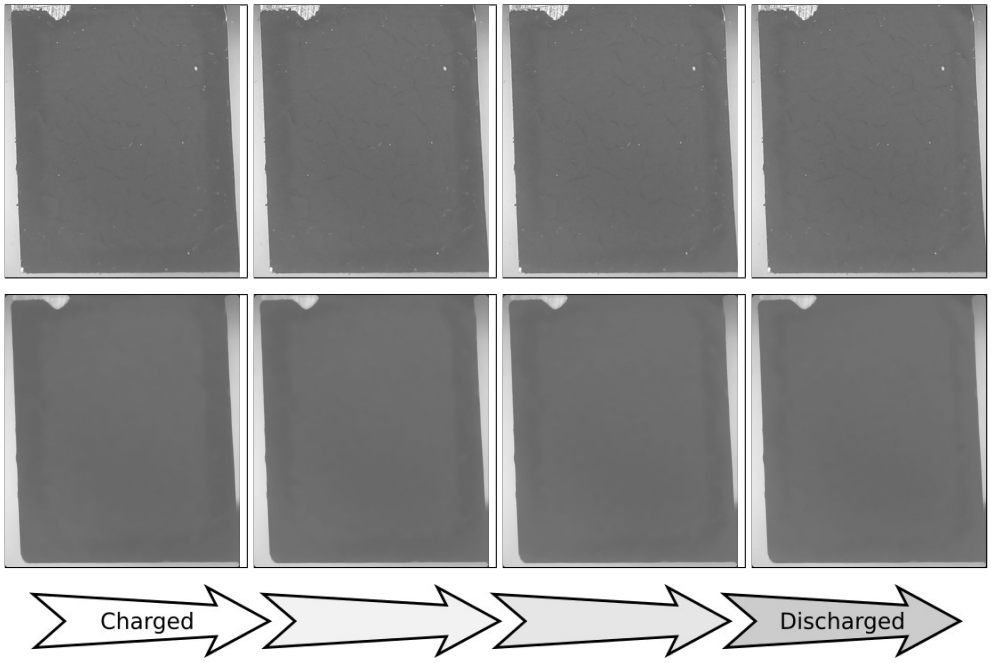



Figure 3.4.: **Top:** Greyscaling example. **Bottom:** Noise reduction. Brightness increased for clarity. Small edges, dots will all disappear after smoothing function.

The effect of the noise reduction is more clear in the following example images. The first image is a binarized image of the original image, the second has the *medfilt2* filter applied to it.

## 3.2. Binarization

In order to find the connected areas in the battery test cell images, the binarization method is implemented. Figure 3.6 shows the result of implementing binarization.

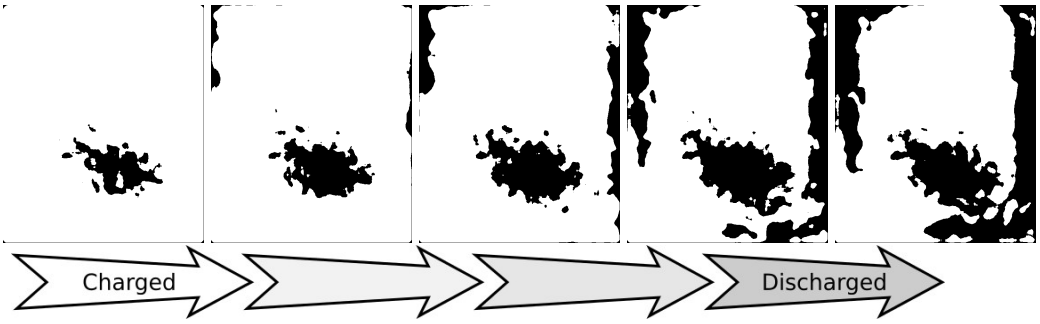



Figure 3.5.: Binarization example.

## 3.3. Edge detection

Implementation of the edge detection method is done through Matlab edge detection on the area of interest. It shows the intensity change through the profile of the battery cell images during charging. Edge detection represent the area which differs during charging and discharging by resulting the below images. This preprocessing step will determine the part which has to be consider for optical flow methods implementations.

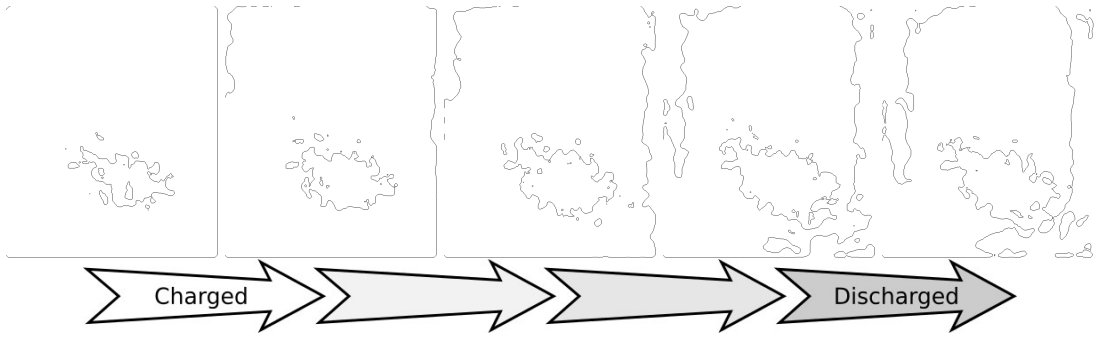



Figure 3.6.: This image shows the edge detection of the battery cell images during charging.

## 3.4. Section difference

This preprocessing step is basically the section difference between the adjacent test cell images. The implemented is done through Matlab coding added into the Matlab code section of this thesis. Figure 3.7 shows the result of the implementation on the gray scaled test cell images.

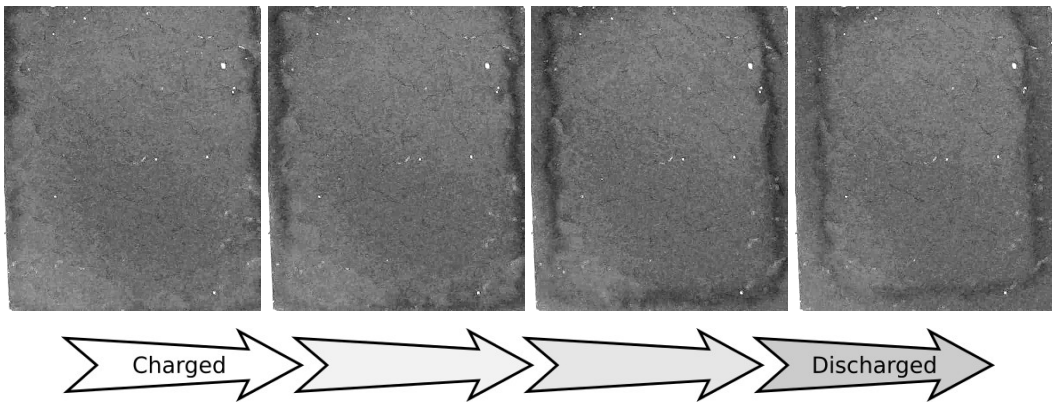



Figure 3.7.: Section difference between the adjacent images.

## 3.5. Denoising

Capturing the battery test cell images and implementing preprocessing methods such as image difference will add some noise to the image, which is not suitable for further use of this result for the optical flow methods. Therefore, one of the preprocessing steps before implementing the optical flow is the method called "De noising" and is done by wiener filter through Mathematica explained in the theory section of denoising. After implementing the denoising method, the noises from the battery cell images are removed but it makes the images blur. The blurring gradient is not analysed. Although the result images are blur compare to the original ones still the pattern of the intensity change during charging and discharging is visible and can be detected properly in the optical flow implementations without major effects. Figure 3.8 shows the result of denoising on the original battery test cell images during the discharging process.

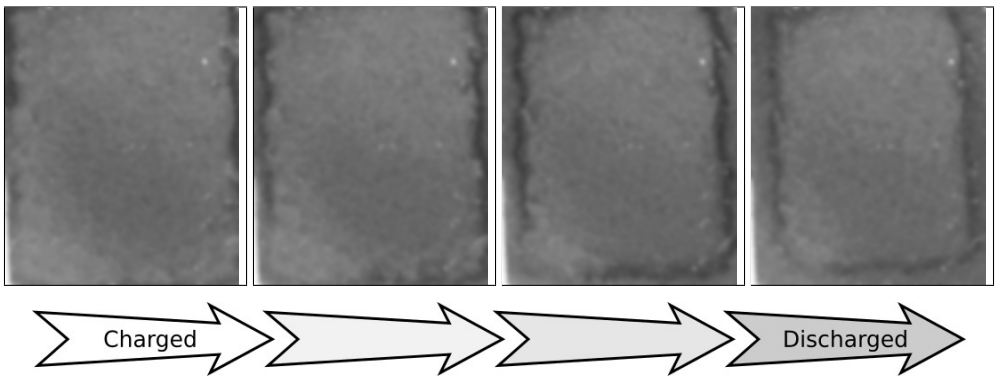



Figure 3.8.: Series of denoised test cell images during the charging process.

## 3.6. Overview of preprocessing steps

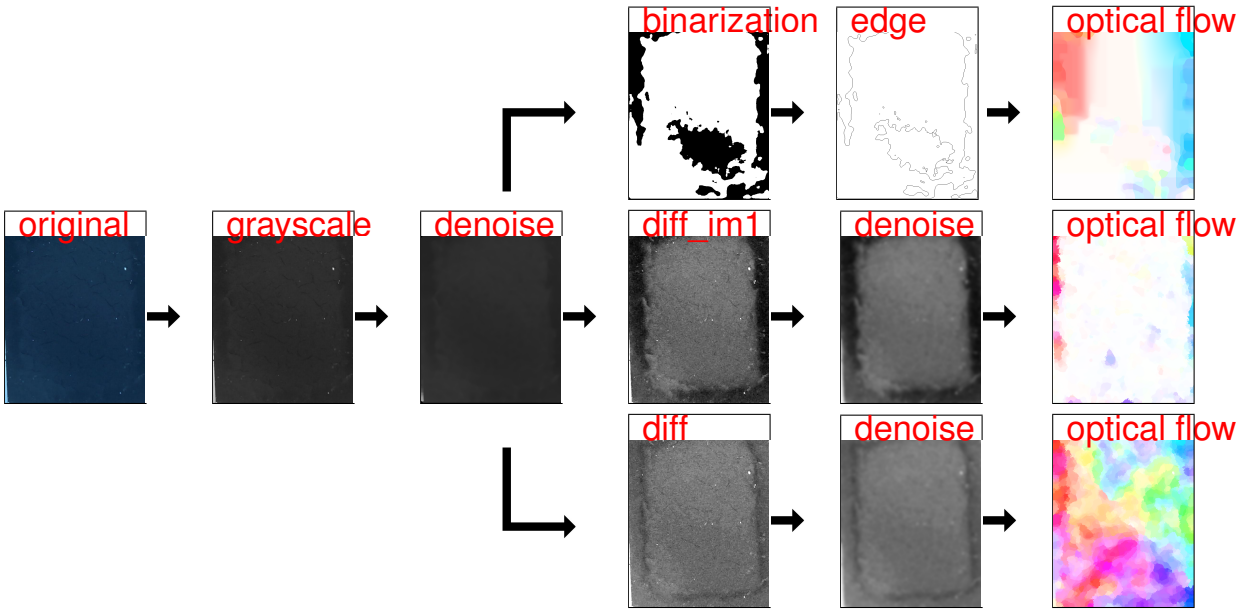



Table 3.1.: This figure presents an overview over the preprocessing steps taken in preparation for the optical flow analysis.

# 4. Optical flow

This chapter will introduce the result of the implementation of the optical flow methods which is explained in the theory part of this thesis. It includes the implementation of optical flow methods on test images in order to find out the constraints and consider possible parameters to obtain better results and then implementation of the optical flow on the preprocessed battery test cell images. The Matlab code which is used for all of the implemented optical flow methods are included in the Matlab code section of this thesis.

## 4.1. Method overview

### 4.1.1. Horn Schunck

This optical flow method which is explained in theory part is divided into three implementations:

- Implementation on test images with background
- Implementation on test images without background
- Implementation on battery test cell image

### 4.1.2. Lucas Kanade

Because it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image. Therefore, the implementation of the Lucas Kanade method is proposed for the further development as one of the optical flow methods.

### 4.1.3. Middlebury

Middlebury test bench mark is a method of the optical flow implementations on the consecutive test images and the battery test cell images which produces Middlebury color maps and vector plots. The vector plot presents the direction of the optical changes through two consecutive test cell images. If there is no optical changes between two images, the plot presents only points. The color map basically presents the direction of optical change by color coding. The advantageous of using color map is that the intensity of optical changes are more visible. If there is no intensity changes, the color map shows white color. The color coding that corresponds to each direction is added in the implementation part of this chapter. The accuracy and reliability of the optical flow calculation algorithms is improved as it is shown in the Middlebury optical flow benchmark. The most precise methods on the Middlebury flow dataset make different options about how to model the objective function, and how to optimize it. Middlebury introduces a baseline algorithm that is "classical", and is a direct descendant of the original Horn Schunck formulation, and then regularly change the model and method by using different techniques from the art. Only small number of key parameters will produce acceptable results [27] [26].

The Matlab code for implementing the Middlebury method is added to the Matlab Code section of this thesis. Result of the implementations are explained and shown in the Middlebury implementation section.

## 4.2. Horn Schunck implementation

### 4.2.1. Horn Schunck on test images

Figure 4.1 and figure 4.2 represent the implementation of the Horn and Schunck method between image 1 and image 2. Image 1 and 2 both has no background and by comparing the image 1 and image 2, it is obvious that the black circles are moving out.

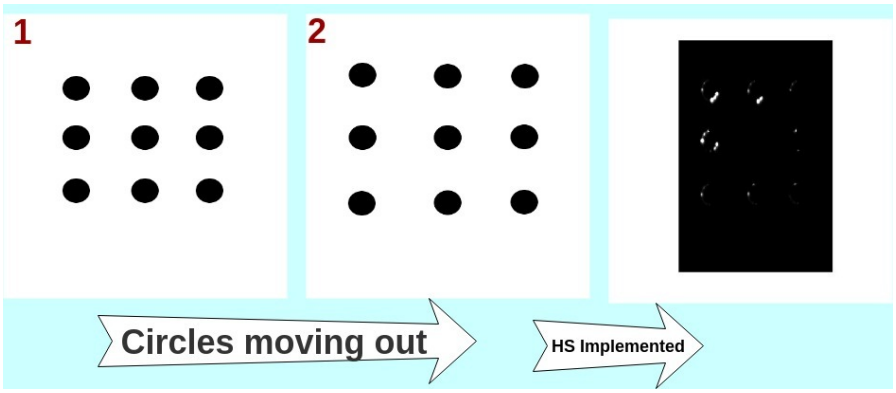



Figure 4.1.: Test image of circles with no background moving out. The Horn and Schunck result doesn't present the movement of the black circles. The solution is to test the result with the the test images which have background in the next part.

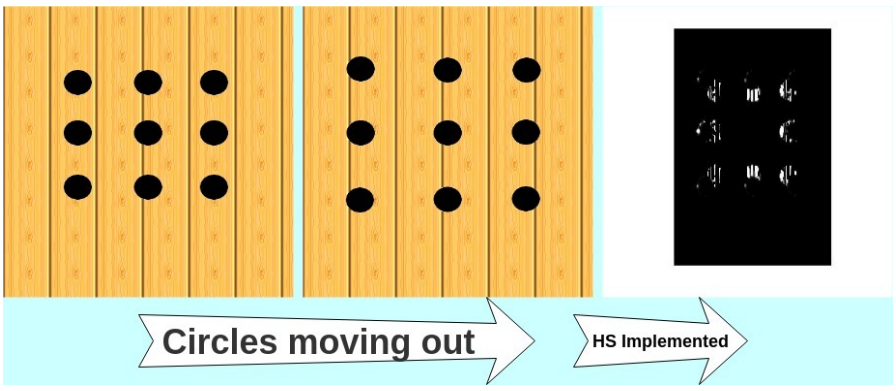



Figure 4.2.: Test image of circles with background moving out. Including the background to the test images will make the Horn and Schunck result more visible but still it doesn't meet the requirements.

### 4.2.2. Horn Schunck on preprocessed battery test cell images

After the implementation of the Horn and Schunck on the test images, the next step is to implement the optimized Matlab code on the battery test cell images which were preprocessed in the previous steps. Result of Horn Schunck method on the denoised images during charging:

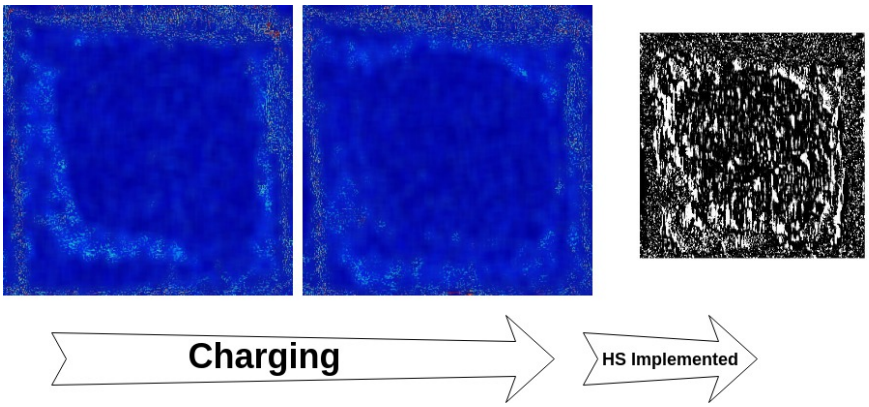



Figure 4.3.: HS method implemented on denoised images during the charging process. The result is not obvious enough although proper preprocessing method was implemented. Therefore, the result is not good enough for observing the optical effects and then visualizing the spatial and temporal distribution of the effect during the charging process. Therefore it is decided to implement the Horn and Schunck method on the other set of preprocessed test cell images which is shown in the next part at figure 4.4.

### 4.2.3. Horn Schunck on preprocessed battery test cell images

The result of the Horn and Schunck method on the other set of preprocessed images was not valid. Therefore, this method is implemented on the preprocessed images(section difference with the adjacent images). Figure 4.4 shows the results of section difference of two adjacent images during the charging process.

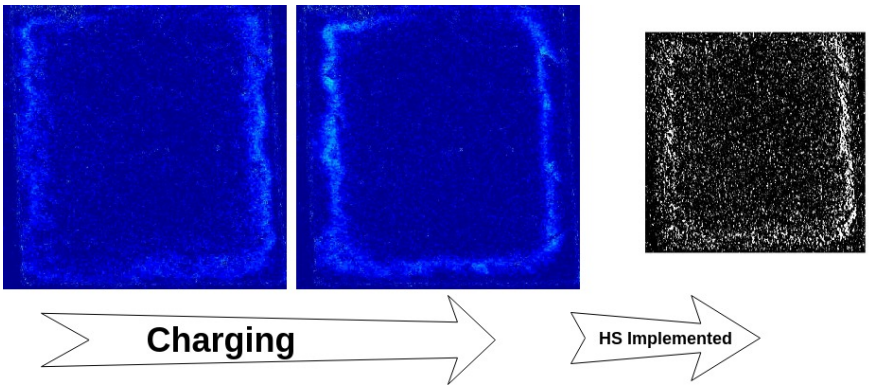



Figure 4.4.: HS method implemented on section difference of two adjacent images during the charging process.

In these series of preprocessed test cell images, the optical changes are more obvious compare to the previous data set but still it doesn't provide accurate information to visualize the spatial and temporal distribution of the effect during the charging and discharging process. Therefore, other optical flow methods which are explained in the next sections are implemented for the further analysis and visualizations of optical effects.

## 4.3. Middlebury implementation

One of the methods of estimating the flow fields which basically resemble the original formulation of Horn and Schunck(HS) is Middlebury optical flow benchmark. This method presents the results by color maps and vector plots. Middlebury optical flow benchmark introduces an algorithm to estimate the flow fields more accurately by optimization methods and modern implementation practices. The classical flow(baseline algorithm), performs better when it is combined with modern optimization and implementation techniques. In this method the Classic+nl-fast and HS methods are used with different possible parameters such as lambda, sigma_d, sigma_s. **Lambda** parameter is a trade-off parameter. The larger it is, the smoother will be the flow fields. **sigmna_d** is parameter of the robust penalty function for the spatial term. **sigma_s** is parameter of the robust penalty function for the data term [26] [27].

Implementation of the Middlebury optical flow benchmark is divide into three parts :

- Implementation on test images with background
- Implementation on test images without background
- Implementation on manipulated battery test cell images
- Implementation on original battery test cell images

The Matlab code which is used for the implementation of the Middlebury optical flow benchmark is included in the Matlab Code section of this thesis.

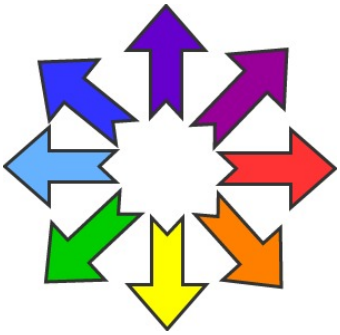


Figure 4.5.: This symbol shows the color coding which corresponds to the direction of the movements in general. Each color represents one unique direction of movement. It is used as the reference to interpret the result of Middlebury.

### 4.3.1. Middlebury on test images

Figure 4.6 and figure 4.7 represent the implementation of Middlebury optical flow benchmark on two test images with and without background. It is obvious that the black circles are moving out.

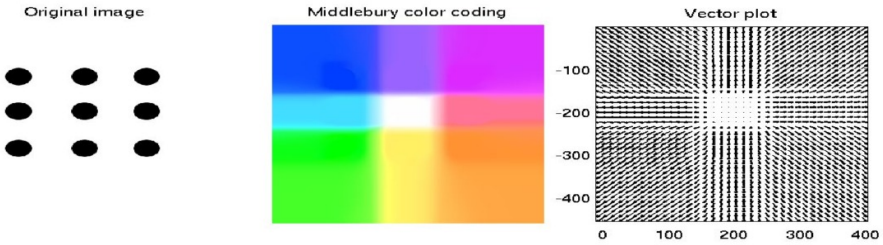



Figure 4.6.: Vector plot and color coding represent the direction of movement of circles without background.


Figure 4.7.: Vector plot and color coding represent the direction of movement of circles with background.

### 4.3.2. Middlebury on manipulated test cell images

A blue stripe is added manually to the right side of test cell images at different positions (to focus on the movement of the blue stripe). The Middlebury vector plot result shows precisely the direction of the movement. This method is used to make sure that the Middlebury method is sensitive to the intensity changes between two images at a particular location at a given period of time.

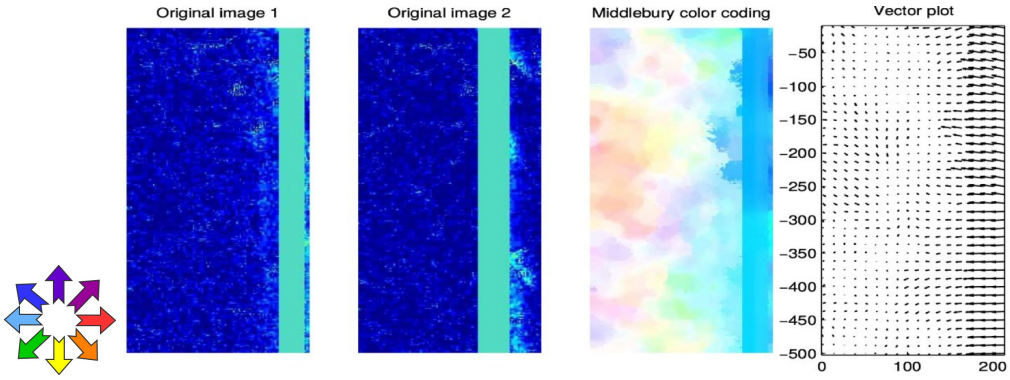



Figure 4.8.: Vector plot and color coding represent the movement of an artificial blue stripe added to the cell images.

### 4.3.3. Middlebury on test cell images

The implementation of the Middlebury flow benchmark on the preprocessed (denoised images) battery test cell images presents the intensity change between these two images.

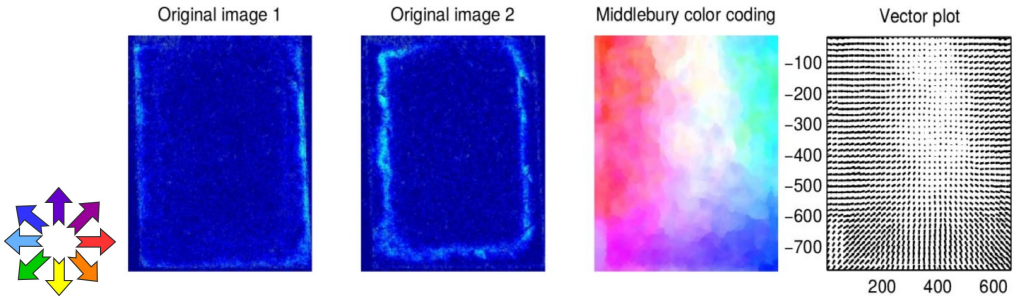



Figure 4.9.: Vector plot and color coding represent the direction of intensity change (diffusion of ion in the battery test cell) between image 1 and image 2 during the discharging process.

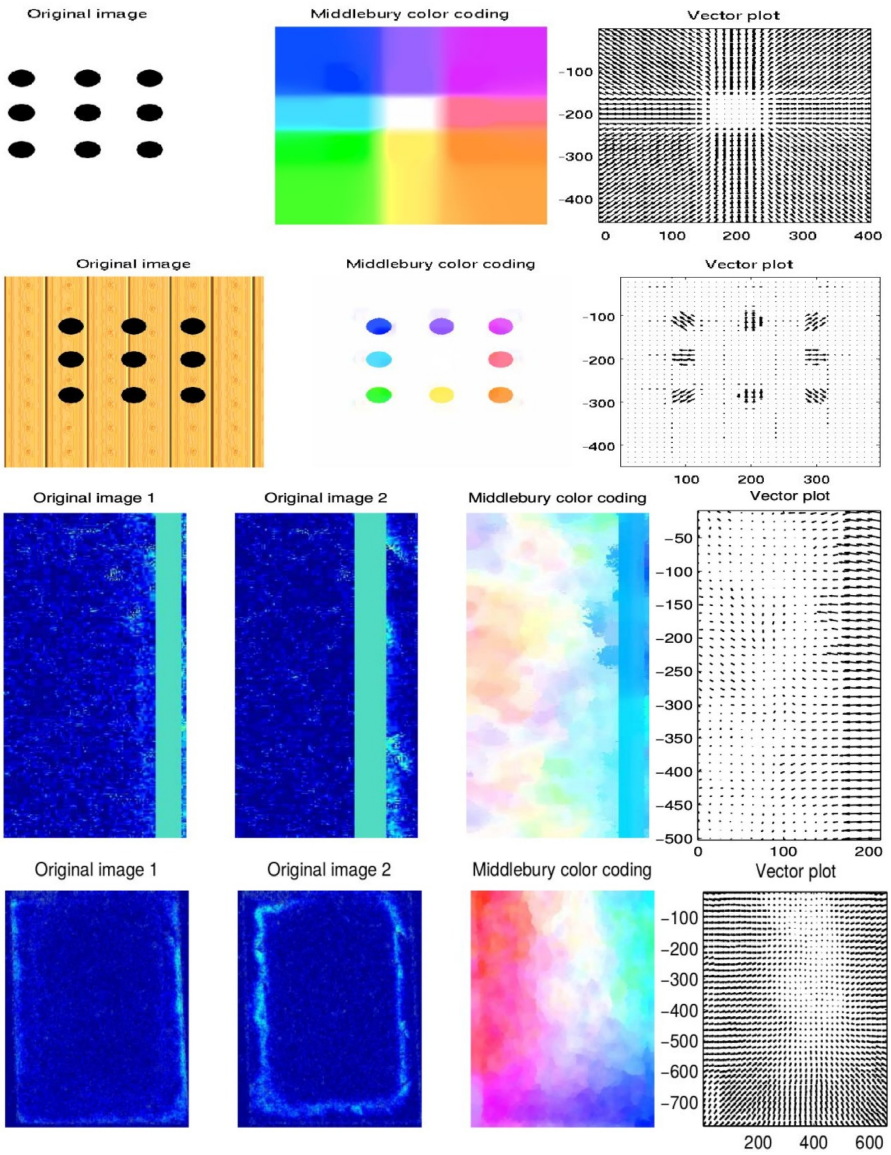



Figure 4.10.: Examples for Middlebury optical implementations of various data sets. **Top:** Expanding circles, interpreted as a movement of the observer. **Second from top:** Expanding circles with background. **Third from top:** Manipulated test cell images with blue stripe. **Fourth from top:** Denoised images of the battery test cells.

# 5. Mass data analysis

This chapter contains the analysis of the battery test cell images of four different data sets. All the information regarding the time between images, number of images and the number of the optical flow results are included in the table 5.1.

## 5.1. Data set information

| **Data set** | MR13Z2P2 | MR17Z2P2 | MR17Z3P1 | MR15Z1P2 |
|---|---|---|---|---|
| **Time between images [min]** | 16:41 | 16:41 | 16:40 | 16:42 |
| **Number of images** | 1181 | 1998 | 146 | 130 |
| **Number of optical flow images** | 14 | 14 | 14 | 14 |

Table 5.1.: Information of various data sets.

For better comparison and analysis of the data sets, mass data analysis is presented as the following steps:

- Presentation of the original test cell image with marked sections for reference, profile and area of interest
- Plots of the electrical measurements of the battery during the charging and discharging process
- Presentation of the preprocessed test cell images
- Presentation of the optical flow results as vector plots and middlebury
- Plot of the intensity profile over distance from electrode border

## 5.2. Analysis steps

### 5.2.1. Preprocessing Matlab files

Table 5.2.1 introduces all of the preprocessing steps which is done on the battery test cell images before the optical flow implementations with their corresponding Matlab code.

Table 5.2.2 introduces the optical methods which is implemented with their corresponding steps and parameters.

| | |
|---|---|
| **A: config.m** | Coordinates of sections<br>Image time stamp<br>Electrical data<br>Image RGB data |
| **B: m01_imageRead_v02.m** | Evaluate profiles and image processing<br>Experimental image manipulations<br>Export profile data<br>Spatial distribution analysis |
| **C: m02_plotCreate.m** | Plot charge, intensity over time for RGB data |
| **D: m03_videoCreate_v02** | Create video out of images and plots<br>Create moving points |
| **E: m00_edge_test** | Section test<br>Binarization<br>Noise removal<br>Section difference<br>Gray scaling<br>Section Edge zero cross<br>Edge detection |
| **F: m00_imagesc_test.m** | Section difference from image_1<br>Edge Test<br>Section binary<br>Gray scaling<br>Section Edge zero cross<br>Edge detection |
| **G: m00_profile_test.m** | Calculate the mean of pixels in the profile<br>Calculate the temporal gradient of the profiles<br>Calculate the spatial gradient within each profile<br>Plot intensity profile over time |

Table 5.2.: Preprocessing steps with their corresponding Matlab file.

### 5.2.2. Optical flow Matlab files

| | |
|---|---|
| **A: Middleburry test bench mark** | Create test image with and without background<br>Plot Middlebury color coding<br>Create vector plot |
| **A_1: Horn Schunck** | Lamda parameter adjustment<br>Sigma s parameter adjustment |
| **A_2: Classic+nl** | Lambda parameter adjustment<br>Sigma S parameter adjustment<br>Sigma D parameter adjustment |

Table 5.3.: Implemented optical flow methods with their corresponding parameters.

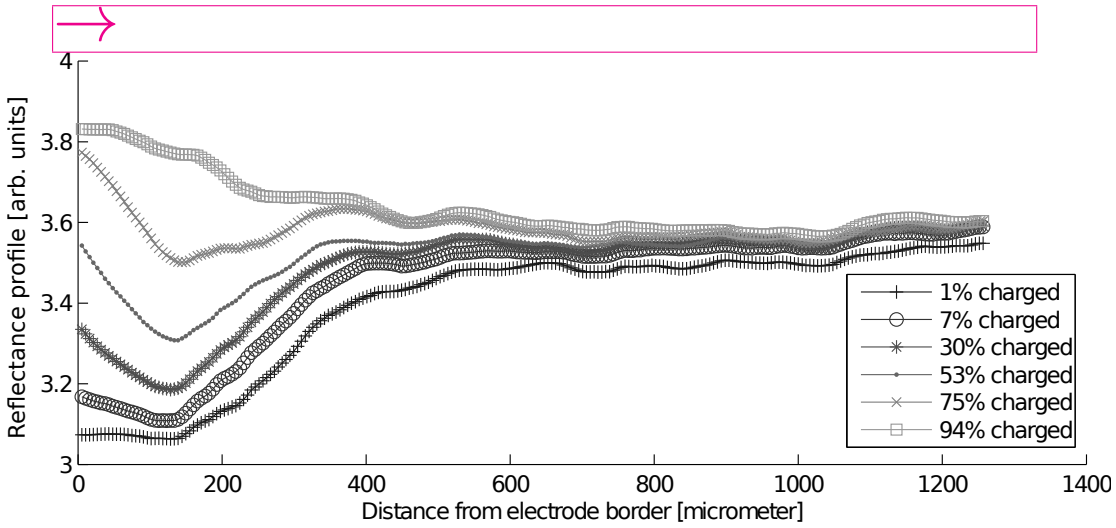



Figure 5.1.: Profiling of the electrode edges proves that relatively only the area which is close to the electrolyte is affected during a single charge. Pink rectangle with the arrow which is parallel to the x axis in the cell area overview images (e.g. figure 5.2) shows the evaluated area. The most obvious change happens only in a small area which is the closest region to the electrode border. The penetration depth is just good enough with the thickness of the typical LFP electrode (200-500 $\mu$m). This process is fully reversible [22].

### 5.2.3. Video creation for data visualization

In order to have an overview and better analysis of the optical effects in parallel with the electrical and optical parameters, the idea of creating videos in Matlab functions are introduced. As it is shown in the table 5.2.1 creating video is one of the primary steps before implementing the optical flow methods. The Matlab code with comments are included in the Matlab code section of this thesis. The video files are added to the CD attached to the bachelor thesis. This chapter will present the snap shots of the video frames produced for each data set. The following Linux command is used together with the Matlab code m03_videoCrate_v02.m to create videos out of the series of images :

avconv -i %d.png -c:v mjpeg ./video.avi

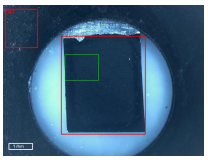
### 5.2.4. Overview of analysis

The results for mass data analysis are presented in a uniform fashion. First, an image of the electrode is shown **(a)**, followed by plots with voltage (blue), current (red), the three RGB channels (red, green, blue), and charge (black) with one of the channels (red, green or blue) **(b)**. A table explains some details of the test run **(c)**. On the next page, results for image preprocessing of test cell images during the discharging process are shown **(d)**, spanning the original image, greyscaling, differential images, binarization and edge detection on the binarized images (see also chapter 3). Finally, Middlebury color plots and vector plots are shown for each set of preprocessed images **(e)**.



**5.1. MR15Z1P2**

a)



b)

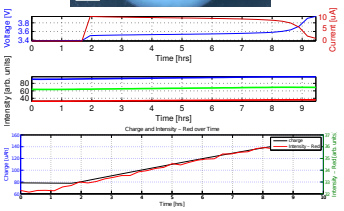



Figure 5.1.: Data for MR13Z2P2.First image shows the test cell image during charging process with the big red rectangle which shows the area of interest and small green rectangle which shows the section profile and the small red rectangle in the corner which shows the area of reference. the graphs presents the electrical and optical parameters of test cell images over the period of the time during the charging process. First graph: Voltage variation. Second graph: Current variation. Third graph: Absolute value of intensity, compensated for RGB channels. Fourth graph: Charge and intensity of red channel.

c)

| Data set | MR15Z1P2 |
|---|---|
| Time between images [min] | 16:42 |
| Number of images | 130 |
| Number of optical flow images | 14 |



d)

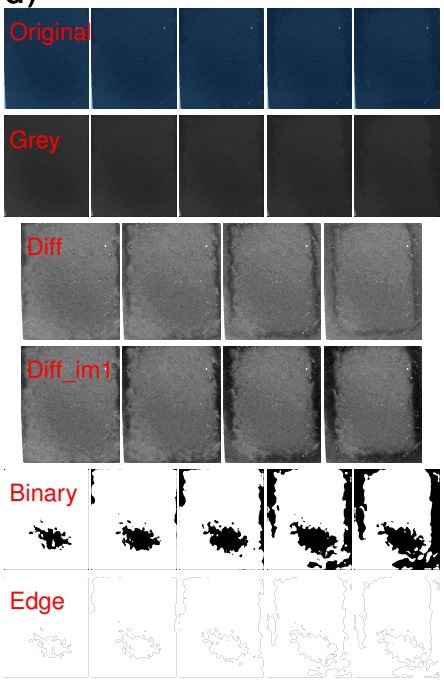



Figure 5.2.: Image preprocessing steps on MR15Z1P2 data set.
First row: selected section from the original test cell image, Second row: Grey scaled of the original image, Third row: The image difference between every image and their neighbor, Fourth row: The image difference between the every image and the last one, Fifth row: Binarized images, Sixth row: Result of edge detections.



e)

**5.1.1. Optical Flow**

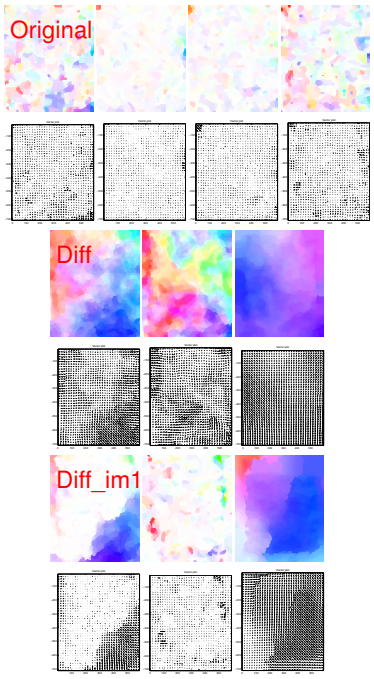



Figure 5.3.: Result of optical flow implementations. First row: Original images as Middlebury

## 5.3. Optical analysis of MR15Z1P2

This is the original dataset used to test preprocessing an optical flow scripts. It shows a complete electrode within the test cell window. During the charge process, the border areas darken considerably. One of the specific problems of this data set is the irregular darkening especially in the lower left corner and the central dark spot in the binarization.

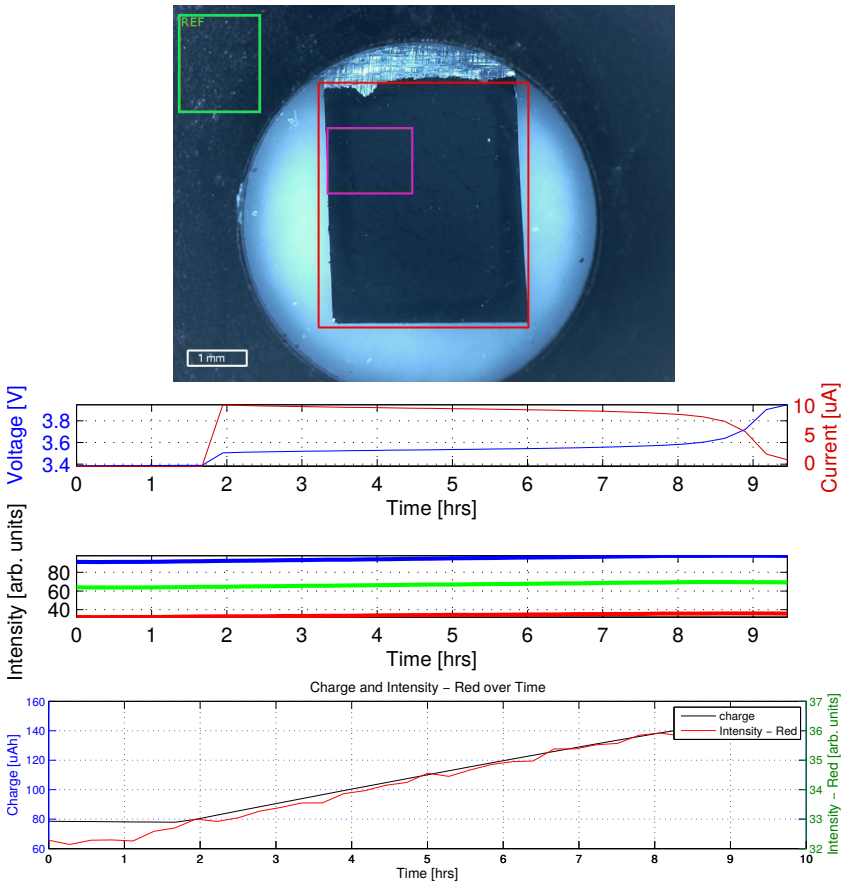



Figure 5.2.: Data for MR15Z1P2. First image shows the test cell image during charging process with the big red rectangle indicating the area of interest and small pink rectangle which shows the section profile with the small green rectangle in the corner which is the reference area. The graphs present the electrical and optical parameters of test cell images over the period of the time during the charging process.

| **Data set** | MR15Z1P2 |
|---|---|
| **Time between images [min]** | 16:42 |
| **Number of images** | 130 |
| **Number of optical flow images** | 14 |

Table 5.4.: MR15Z1P2 Data set information

### 5.3.1. MR15Z1P2 Preprocessing

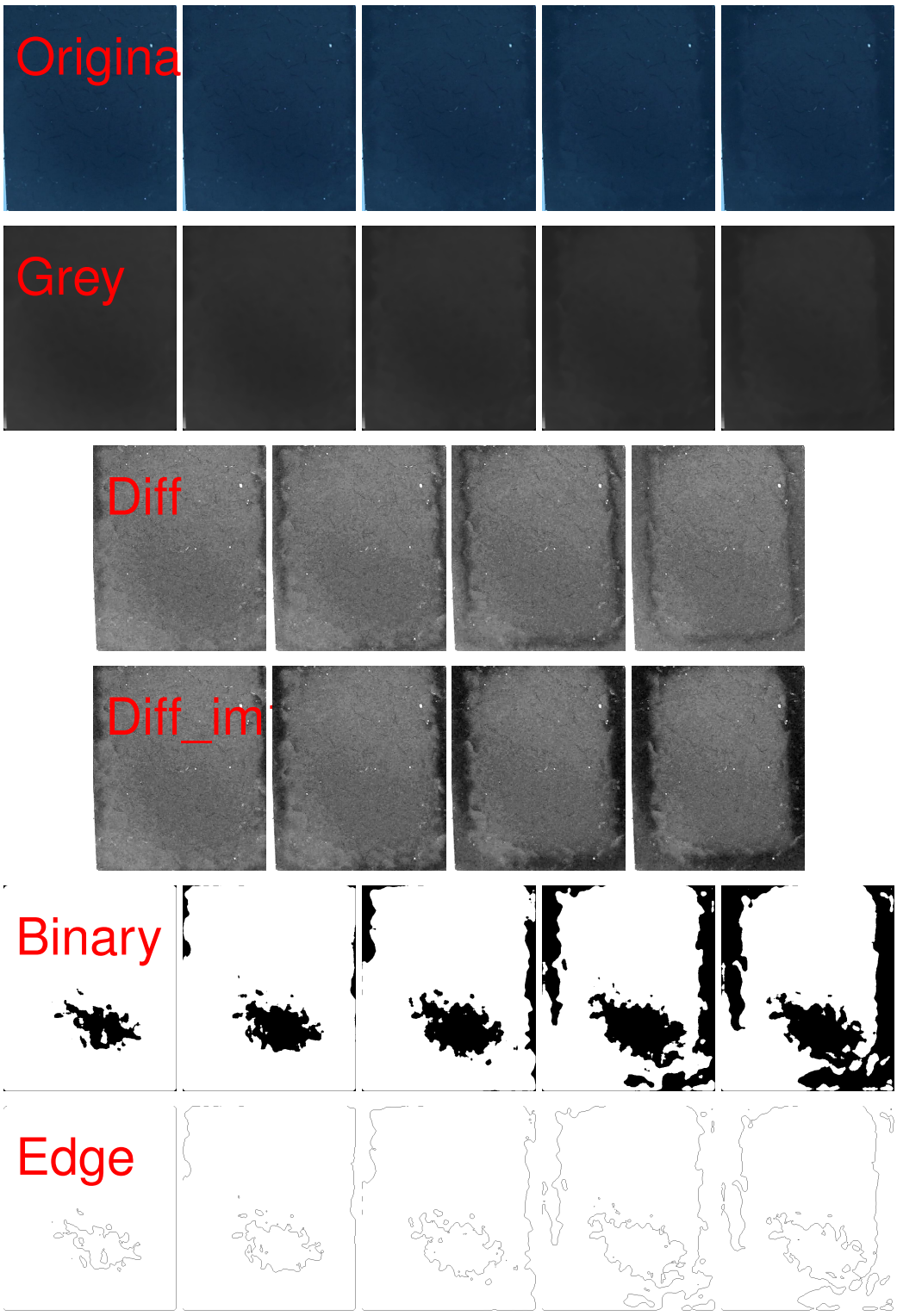



Figure 5.3.: Image preprocessing steps on MR15Z1P2 data set.
**Original:** Selected section from the original test cell image. **Grey:** Grey scaled of the original image. **Diff**: The image difference between every image and their neighbor. **Diff_im**: The image difference between the every image and the last one, followed by the binarized and edge detected results.

### 5.3.2. MR15Z1P2 Optical flow

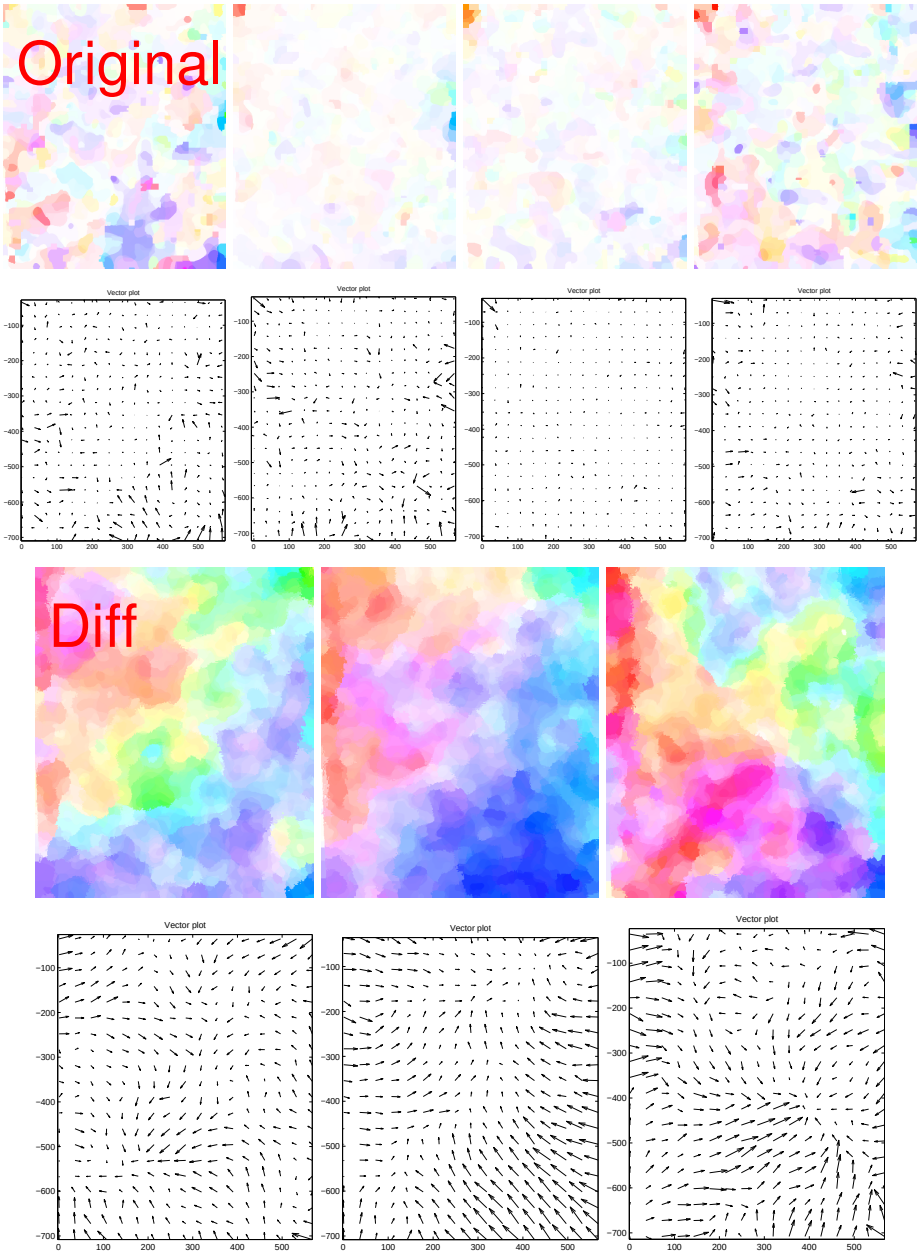



Figure 5.4.: First part of optical flow implementation on the MR15Z1P2 preprocessed images as the Middlebury and vector plot representation. **Original**: Original cell images as Middlebury with the corresponding vector plot in the second row. **Diff**: The image difference between every image and their neighbor as Middlebury with the corresponding vector plot in the fourth row.

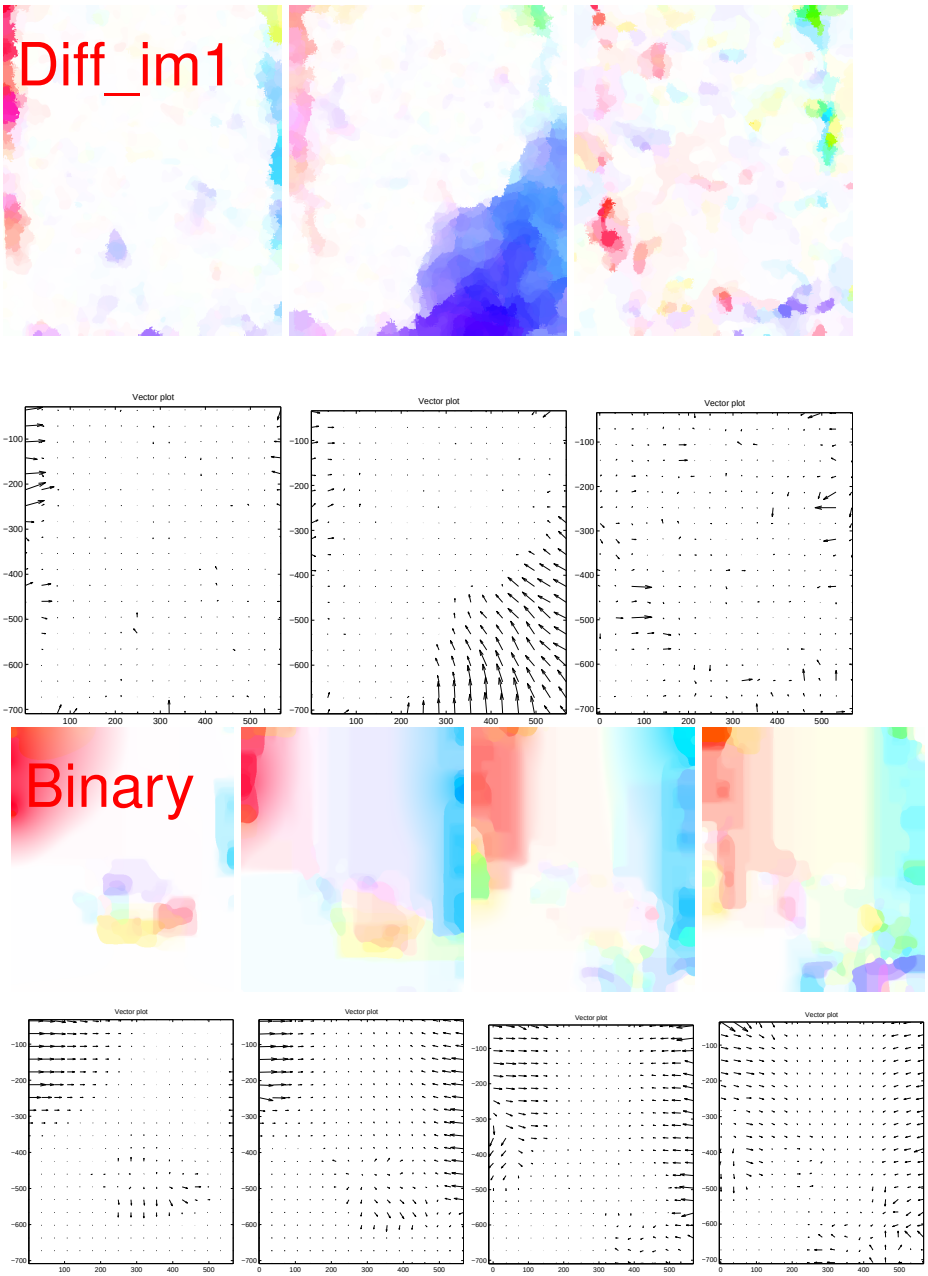



Figure 5.5.: Second part of optical flow on the MR15Z1P2 preprocessed images as Middlebury and vector plot representation. **Diff_im**: The image difference between the every image and the last one as Middlebury with the corresponding vector plot in the second row. **Binary**: Binarized images as Middlebury with the corresponding vector plot in the fourth row.

### 5.3.3. MR15Z1P2 Intensity profile

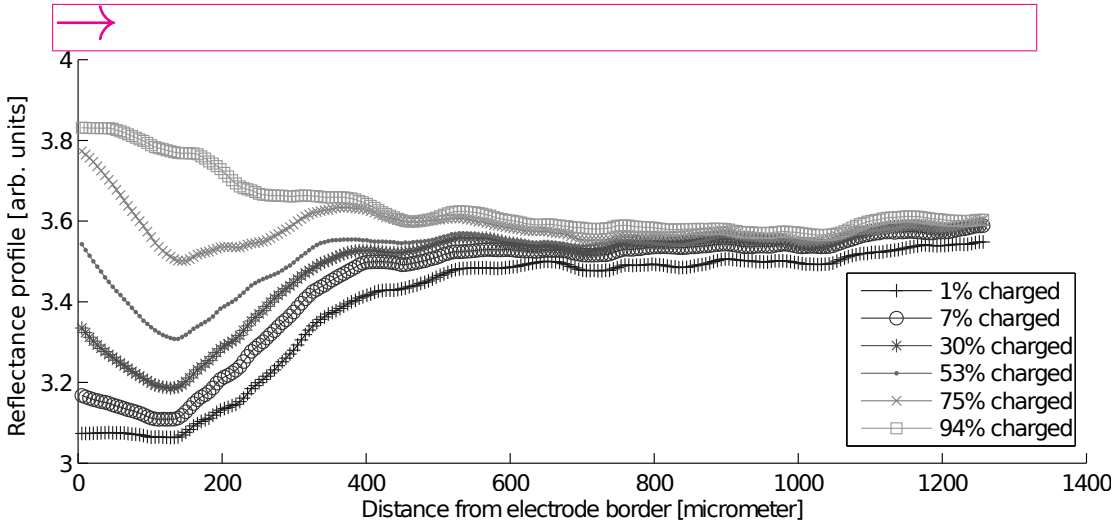



Figure 5.6.: Reflectance profile over the distance from the electrode border. Profiling of the electrode edges proves that mainly only the area which is near the electrolyte is afflicted during a single charge. The mentioned evaluated area is marked with the pink rectangle shown in figure 5.2 and the arrow is parallel to the x axis. The main changes happen only in a short region closest to the electrode border.

### 5.3.4. MR15Z1P2 Video visualization

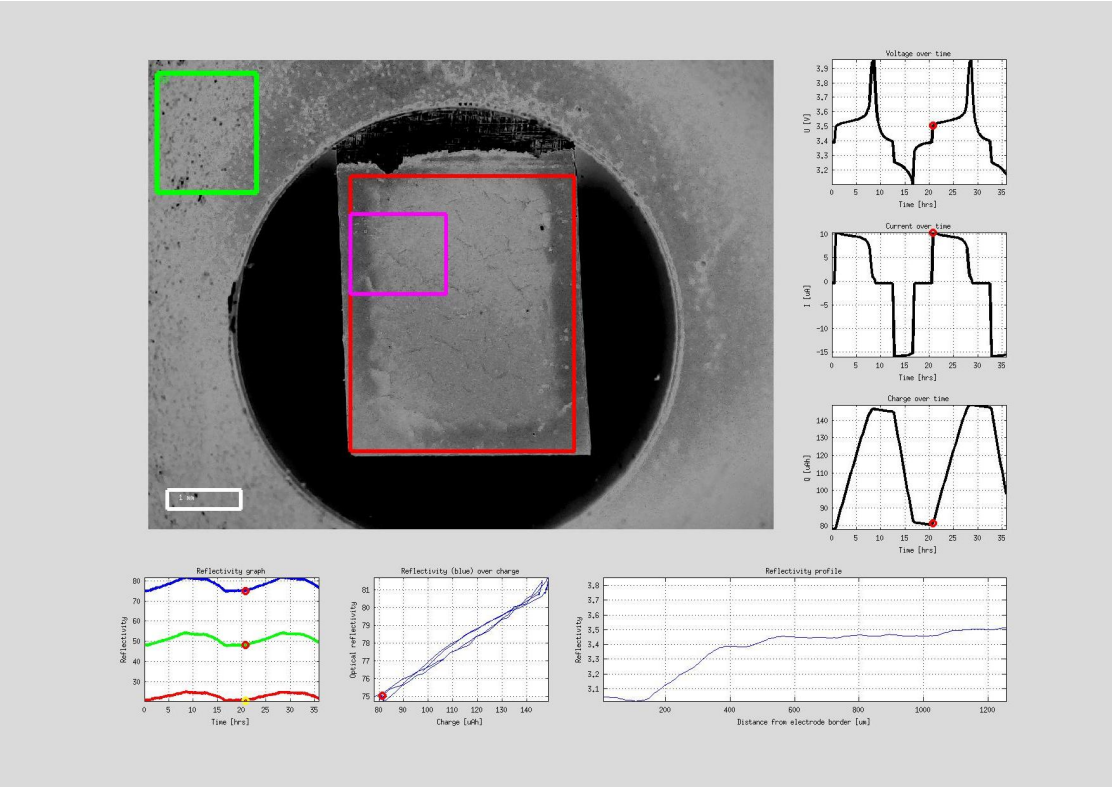



Figure 5.7.: Video frame MR15Z1P2 data set . Test cell Greyscaled images during charging and discharging process with the big red rectangle which shows the area of interest and small pink rectangle which shows the section profile and the small green rectangle which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

## 5.4. Optical analysis of MR17Z3P1

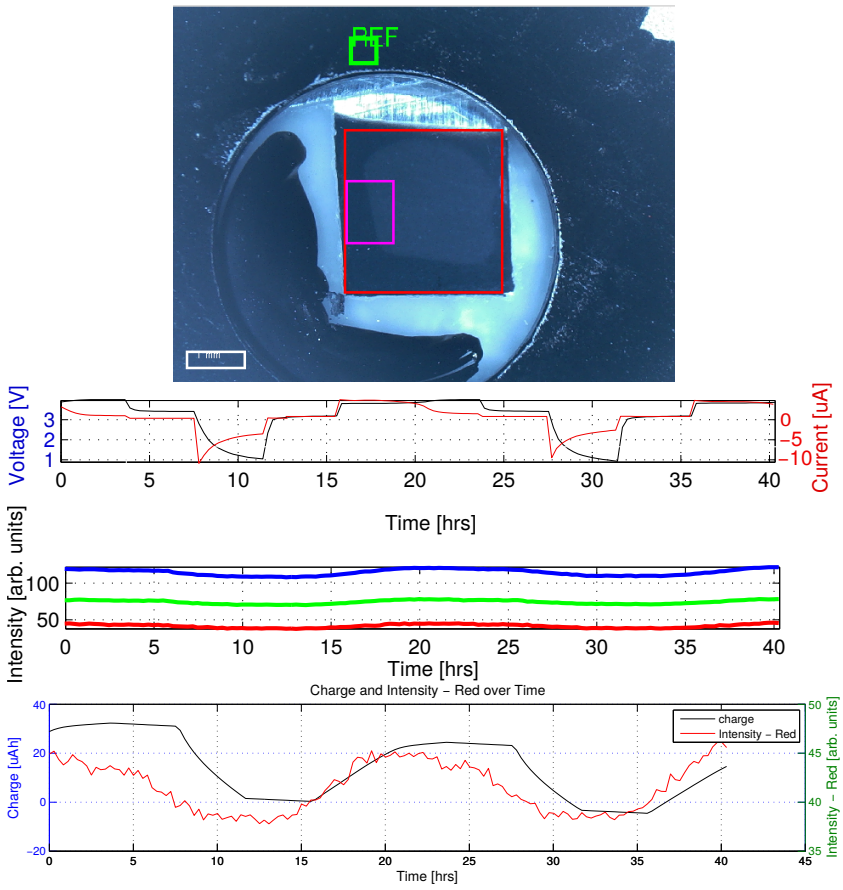



Figure 5.8.: Data for MR17Z3P1. First image shows the test cell image with the big red rectangle which shows the area of interest and the pink rectangle at the left side which shows the section profile and the small green rectangle at the top which shows the reference area. The graphs present the electrical and optical parameters of test cell images over the period of the time.

| **Data set** | MR17Z3P1 |
|---|---|
| **Time between images [min]** | 16:40 |
| **Number of images** | 146 |
| **Number of optical flow images** | 14 |

Table 5.5.: MR17Z3P1 Data set information

### 5.4.1. MR17Z3P1 Preprocessing

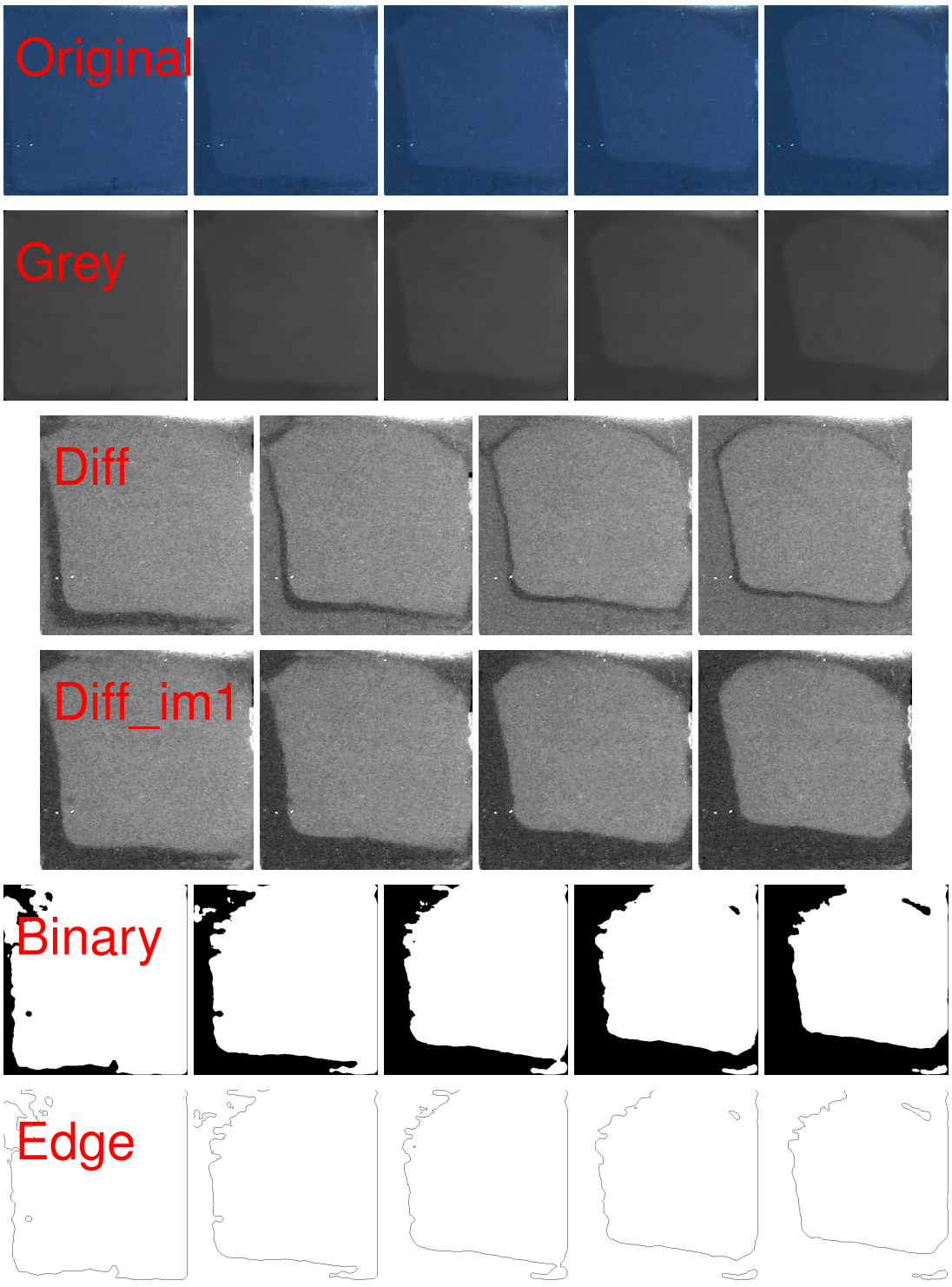



Figure 5.9.: Image preprocessing steps on MR17Z3P1 data set. **Original:** Selected section from the original test cell image. **Grey:** Grey scaled of the original image. **Diff**: The image difference between every image and their neighbor. **Diff_im**: The image difference between the every image and the last one, followed by the binarized and edge detected results.

### 5.4.2. MR17Z3P1 Optical flow

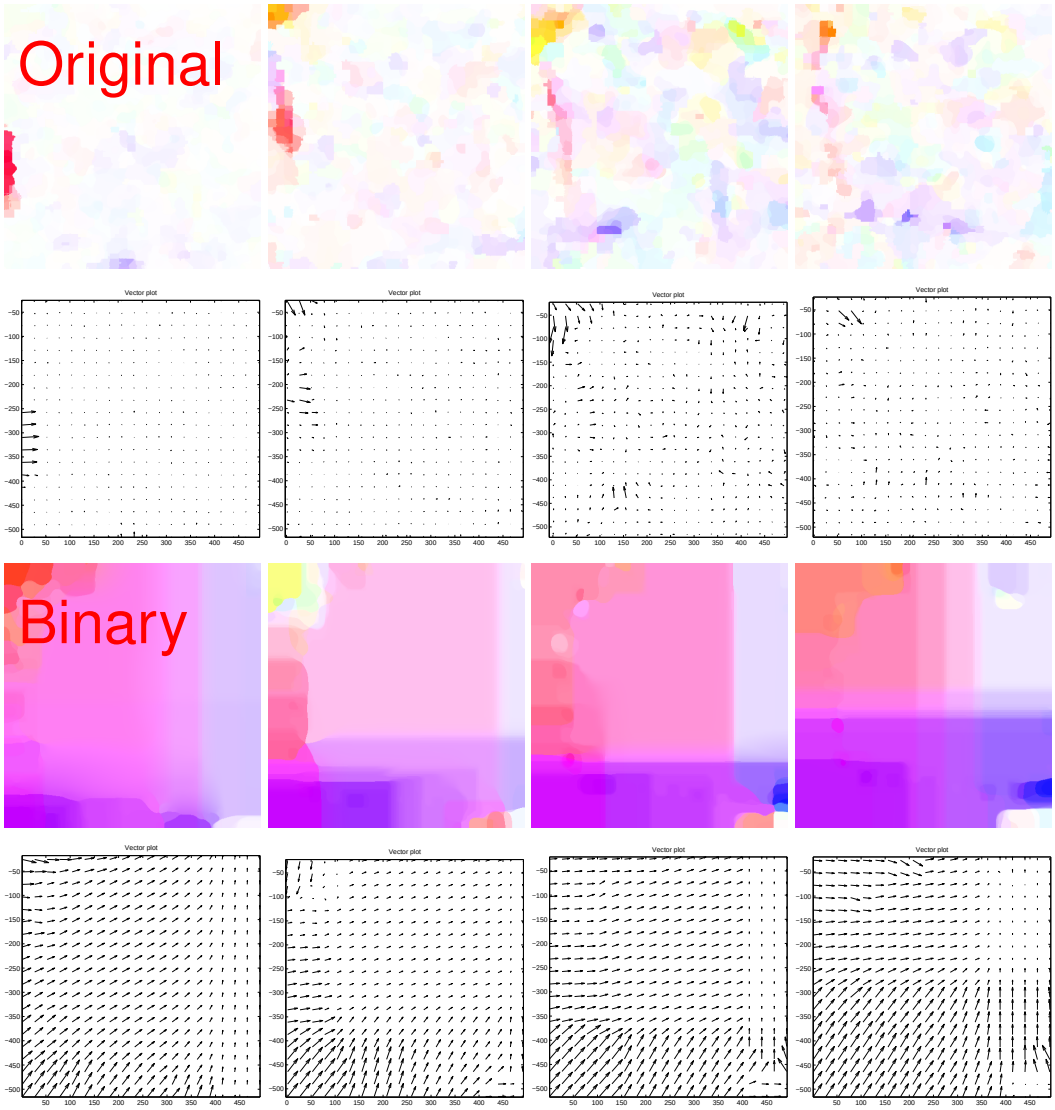



Figure 5.10.: First part of optical flow implementation on the MR17Z3P1 preprocessed images as the middlebury and vector plot representation. **Original**: Original cell images as Middlebury with the corresponding vector plot in the second row. **Binary**: Binarized images as Middlebury with the corresponding vector plot in the fourth row.

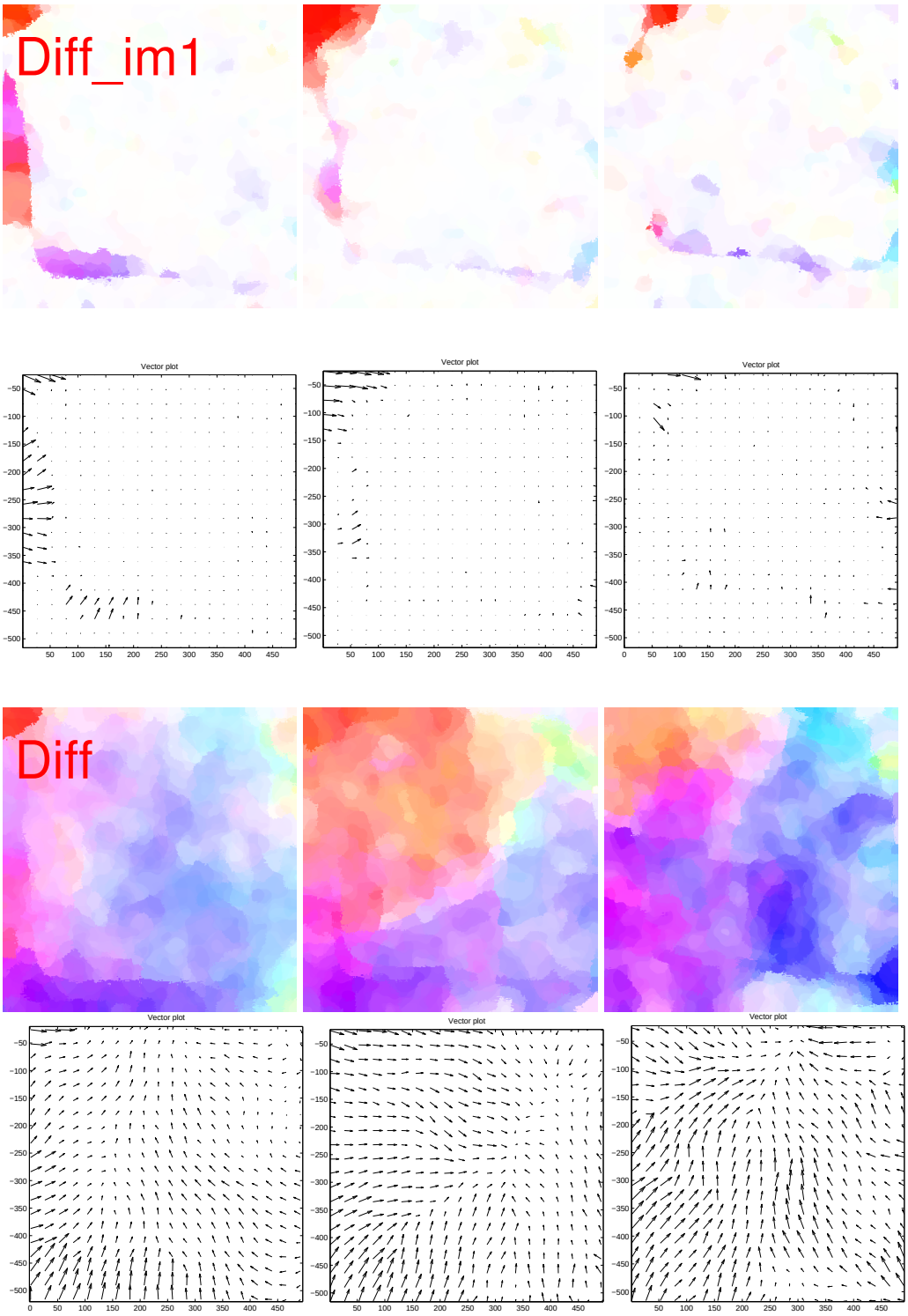



Figure 5.11.: Second part of optical flow on the MR17Z3P1 preprocessed images as Middlebury and vector plot representation. **Diff_im**: The image difference between the every image and the last one as Middlebury with the corresponding vector plot in the second row. **Diff**: The image difference between every image and their neighbor as Middlebury with the corresponding vector plot in the fourth row.

### 5.4.3. MR17Z3P1 Intensity profile

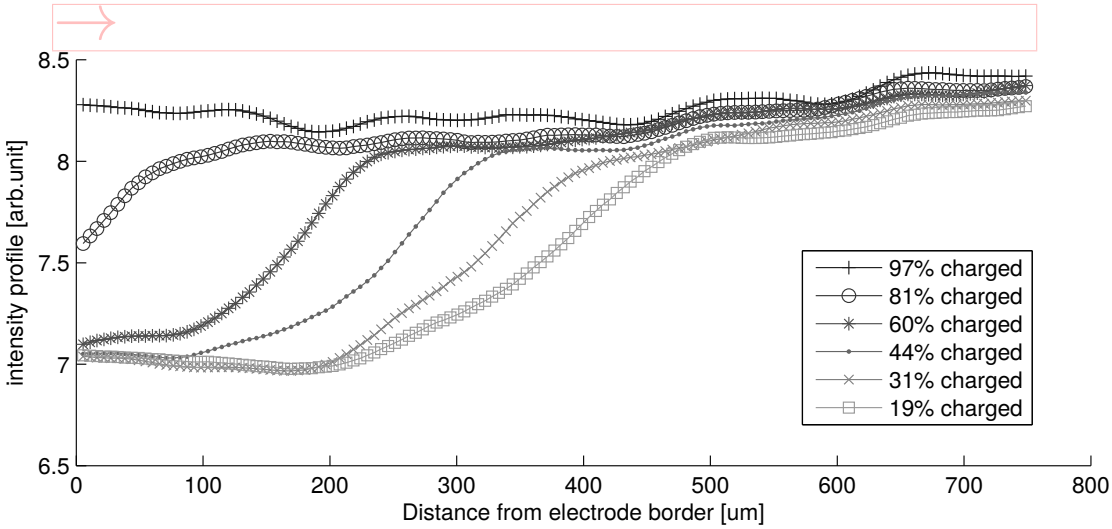



Figure 5.12.: Reflectance profile over the distance from the electrode border. Profiling of the electrode edges proves that mainly only the area which is near the electrolyte is afflicted during a single charge. The mentioned evaluated area is marked with the pink rectangle shown in figure 5.8 and the arrow is parallel to the x axis. The main changes happen only in a short region closest to the electrode border.

### 5.4.4. MR17Z3P1 Video visualization

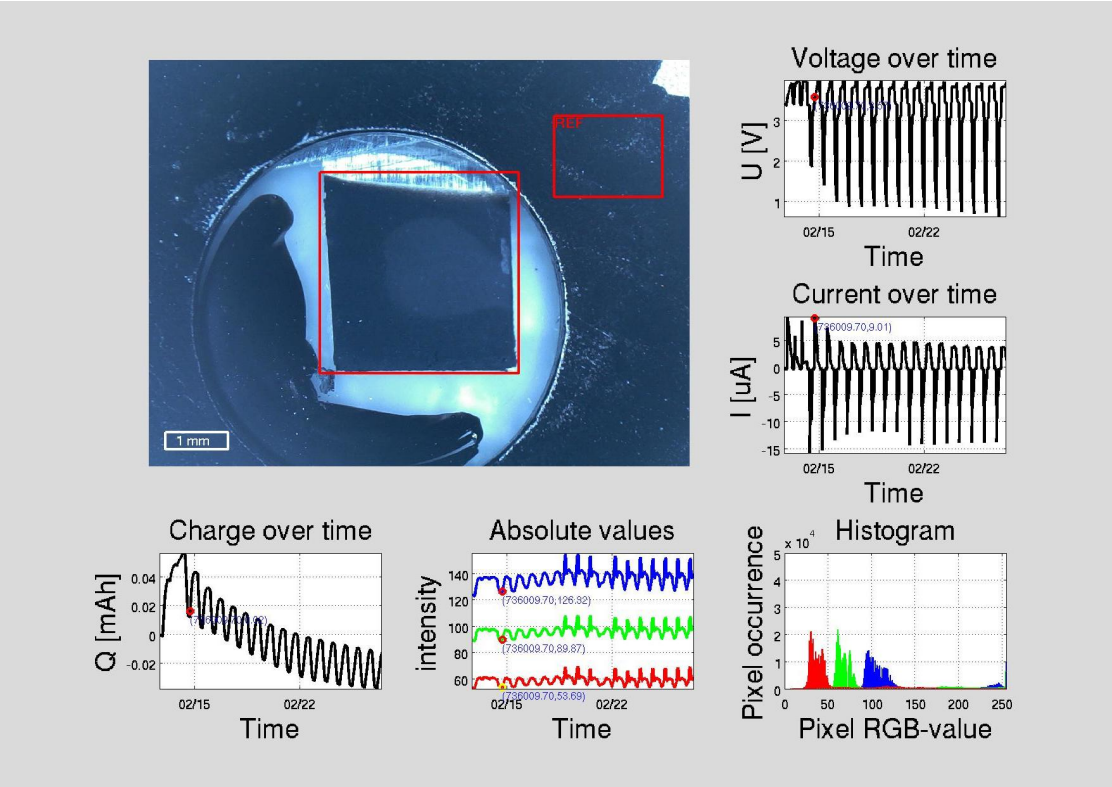



Figure 5.13.: Video frame MR17Z3P1 data set . Test cell images during charging and discharging process with the big red rectangle which shows the area of interest and smaller red rectangle which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

## 5.5. Optical analysis of MR13Z2P2

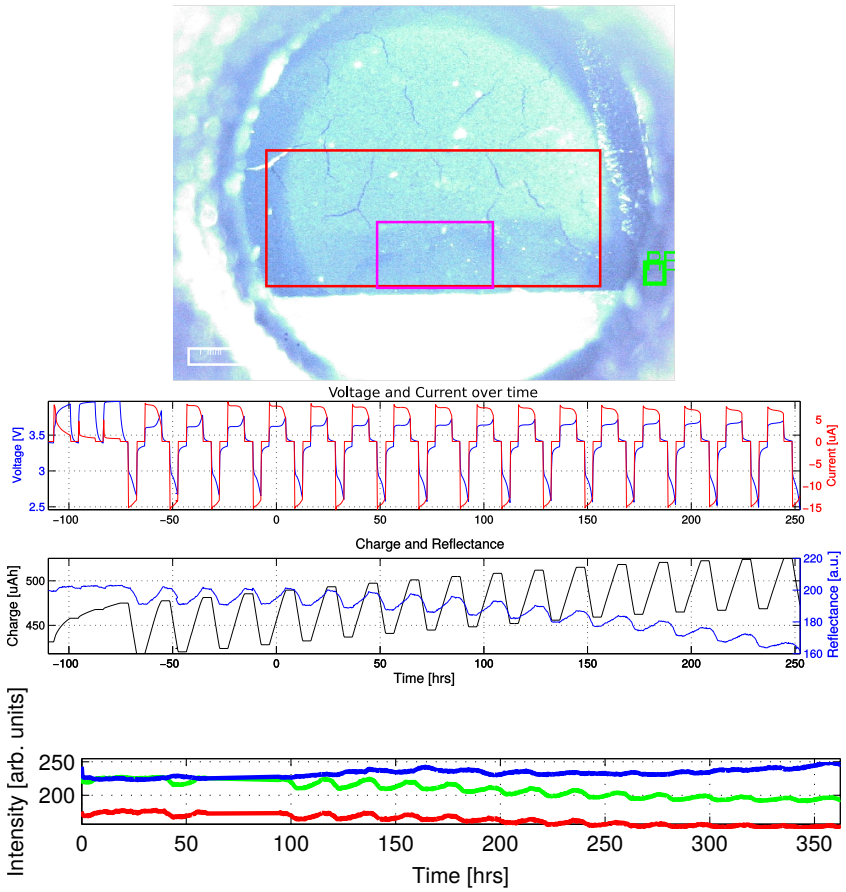



Figure 5.14.: Data for MR13Z2P2. First image: The test cell image with the big red rectangle as the area of interest, pink rectangle as the section profile, green rectangle as the reference area. First graph: Voltage variation together with current variation. Second graph: Charge and reflectance. Third graph: Absolute value of intensity, compensated for RGB channels.

| **Data set** | MR13Z2P2 |
|---|---|
| **Time between images [min]** | 16:41 |
| **Number of images** | 1181 |
| **Number of optical flow images** | 14 |

Table 5.6.: MR13Z2P2 Data set information

### 5.5.1. MR13Z2P2 Preprocessing

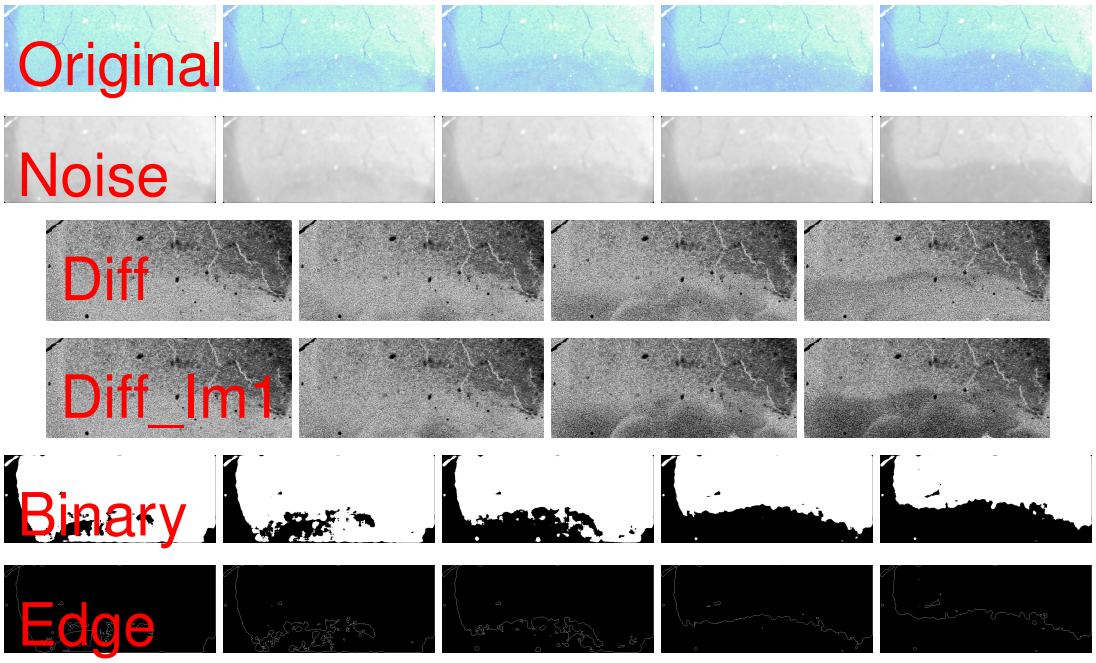



Figure 5.15.: Image preprocessing steps on MR13Z2P2 data set. **Original :** Selected section from the original test cell image. **Noise:** Noise reduction of the original image. **Diff:** The image difference between every image and their neighbor. **Diff_im1:** The image difference between the every image and the last one, followed by the binarized and edge detected results.

### 5.5.2. MR13Z2P2 Optical flow

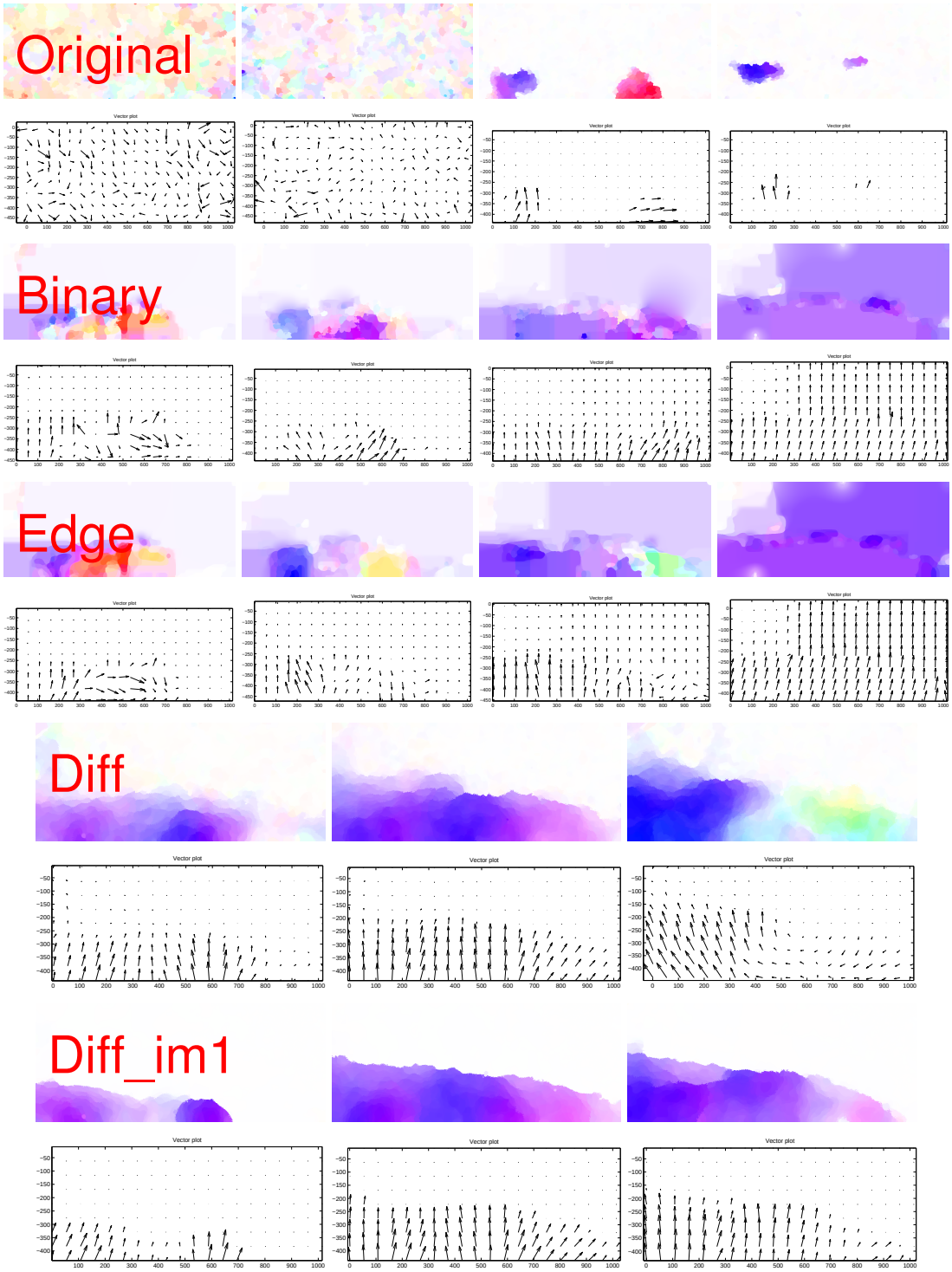



Figure 5.16.: Result of optical flow on the MR13Z2P2 preprocessed images as Middlebury and vector plot representation. **Original**: Original cell images as Middlebury with the corresponding vector plot in the second row. **Binary**: Binarized images as Middlebury with the corresponding vector plot in the fourth row. **Edge**: Edge detected images as Middlebury with the corresponding vector plot in the sixth row. **Diff**: The image difference between every image and their neighbor as Middlebury with the corresponding vector plot in the eight row. **Diff_im**: The image difference between the every image and the last one as Middlebury with vector plot.

### 5.5.3. MR13Z2P2 Intensity profile

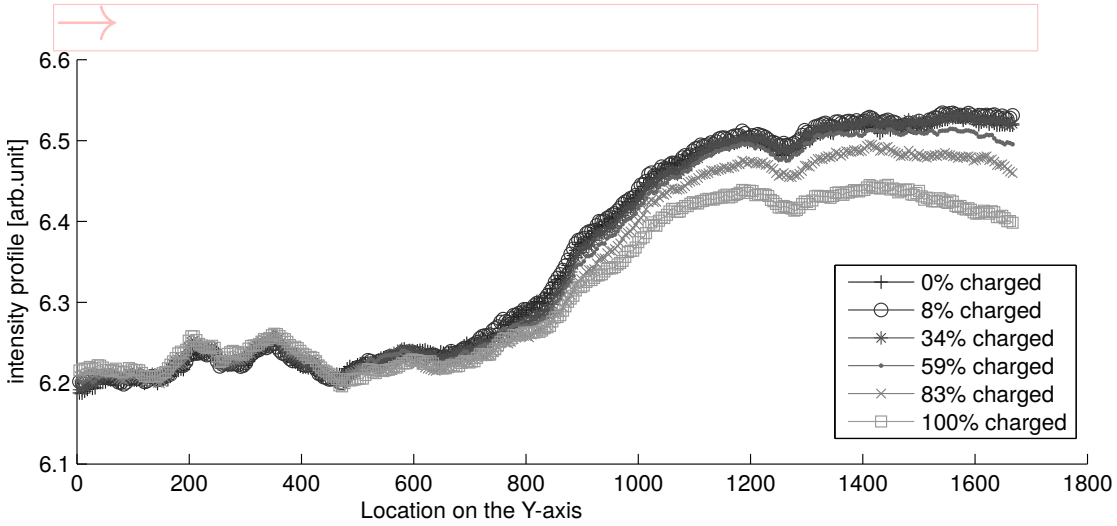



Figure 5.17.: Reflectance profile over the distance from the electrode border. The mentioned evaluated area is marked with the pink rectangle shown in figure 5.14 and the arrow is parallel to the x axis. The main changes happen only in a short region closest to the electrode border. The area is evaluated vertically.

### 5.5.4. MR13Z2P2 Video visualization

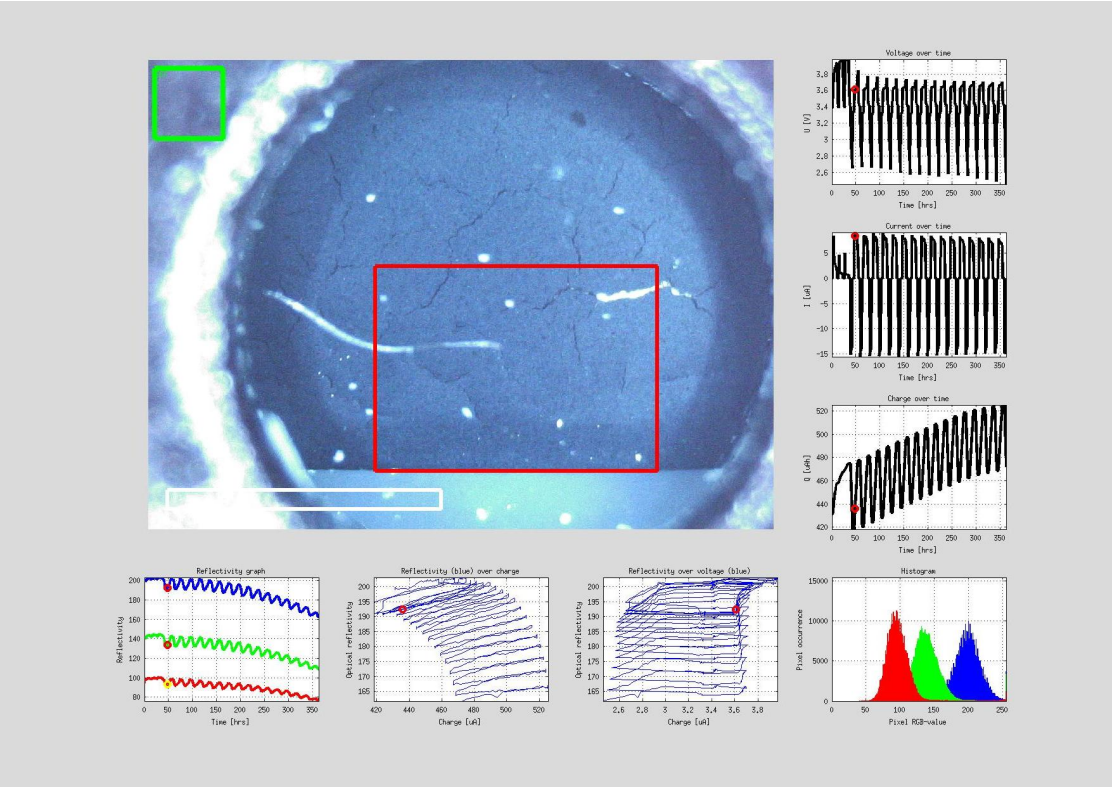



Figure 5.18.: Video frame MR13Z2P2 data set . Test cell images during charging and discharging process with the big red rectangle which shows the area of interest and small green square which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

## 5.6. Optical analysis of MR17Z2P2

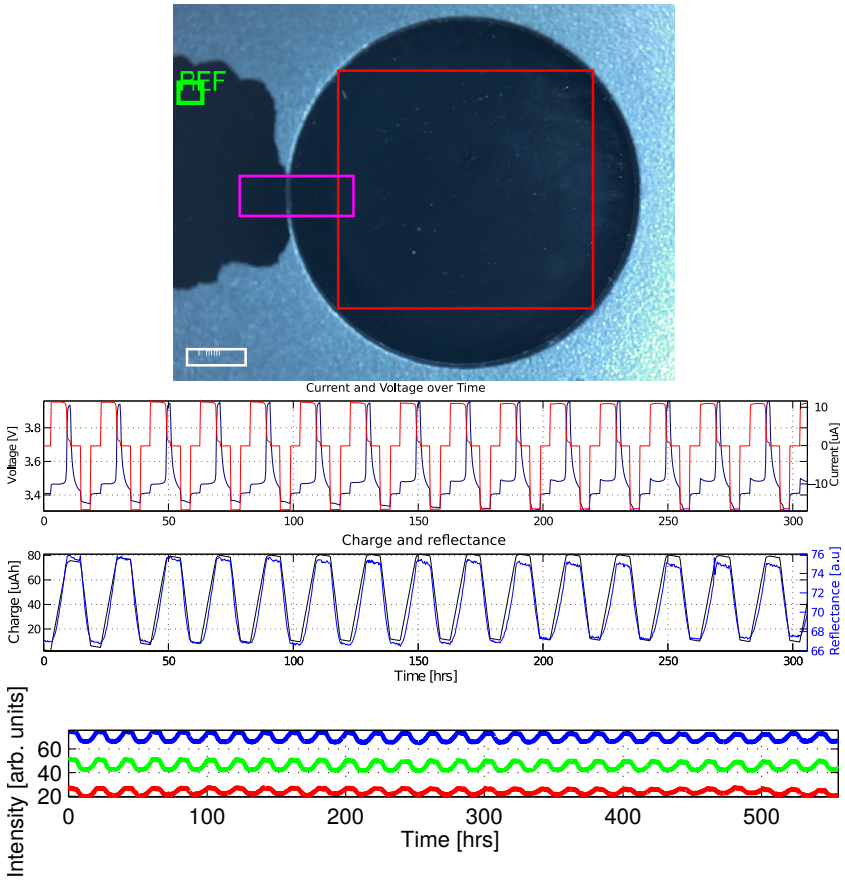



Figure 5.19.: First image: The test cell image with the red rectangle as the area of interest, pink rectangle as the section profile, green rectangle as the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of the time. First graph: Voltage and current variation. Second graph: Charge and reflectance. Third graph: Absolute value of intensity, compensated for RGB channels.

| **Data set** | MR17Z2P2 |
|---|---|
| **Time between images [min]** | 16:41 |
| **Number of images** | 1998 |
| **Number of optical flow images** | 14 |

Table 5.7.: MR17Z2P2 Data set information

### 5.6.1. MR17Z2P2 Preprocessing

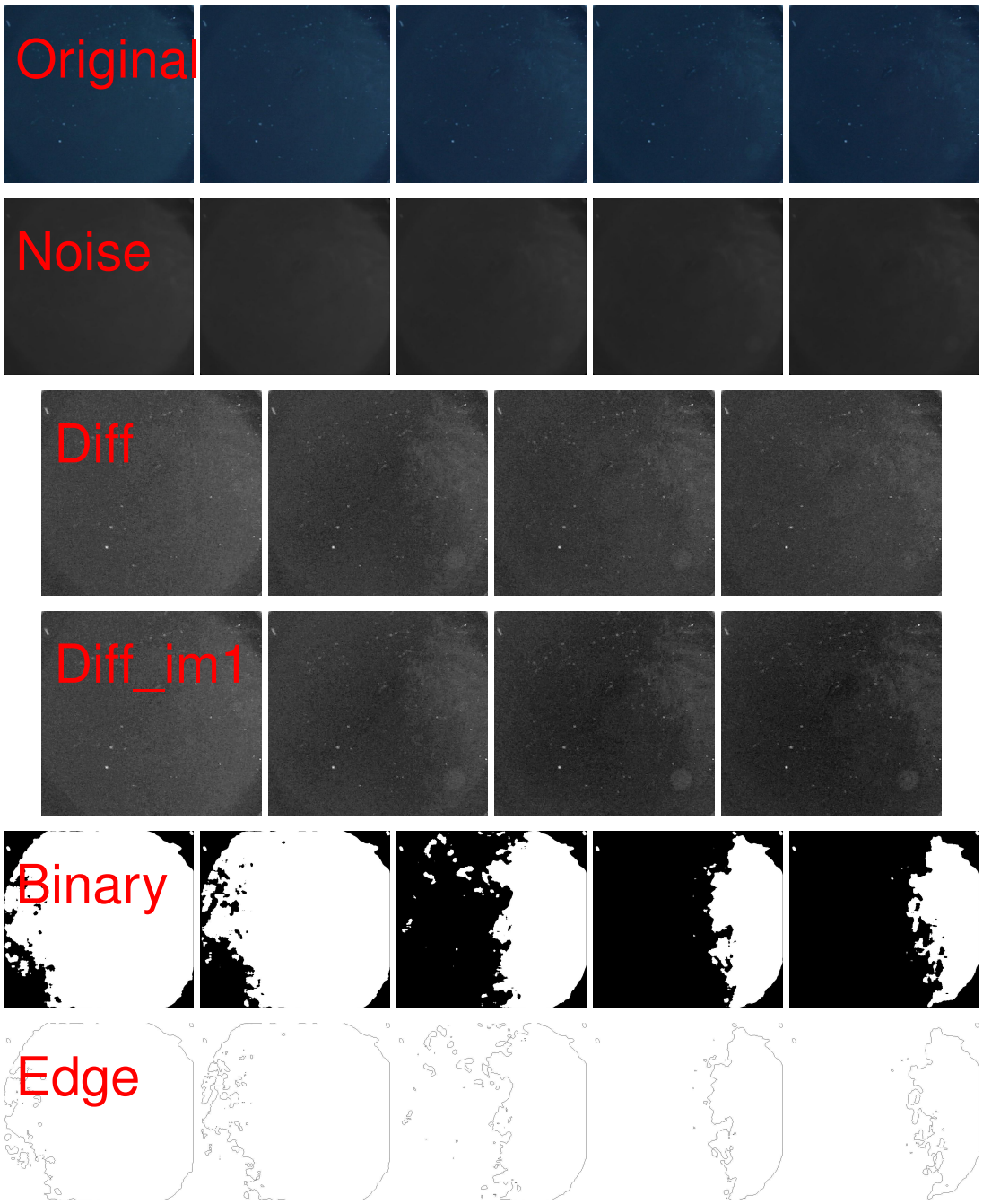



Figure 5.20.: Image preprocessing steps on MR17Z2P2 data set. **Original :** Selected section from the original test cell image. **Noise:** Noise reduction of the original image. **Diff:** The image difference between every image and their neighbor. **Diff_im1:** The image difference between the every image and the last one, followed by the binarized and edge detected results.

### 5.6.2. MR17Z2P2 Optical flow

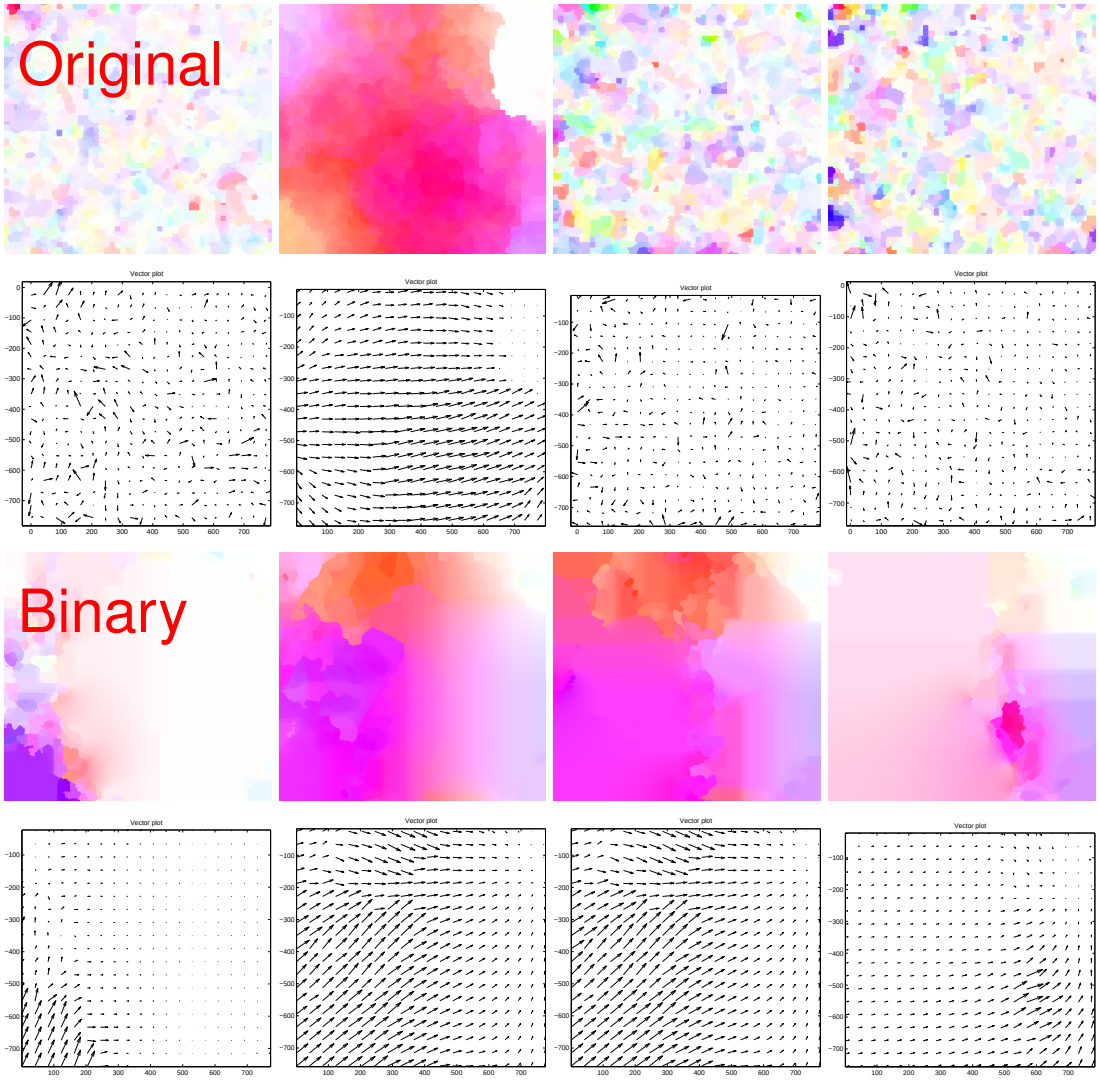



Figure 5.21.: First part of optical flow implementations on the MR17Z2P2 preprocessed images as Middlebury and vector plot representation. **Original**: Original cell images as Middlebury with the corresponding vector plot in the second row. **Binary**: Binarized images as Middlebury with the corresponding vector plot in the fourth row.

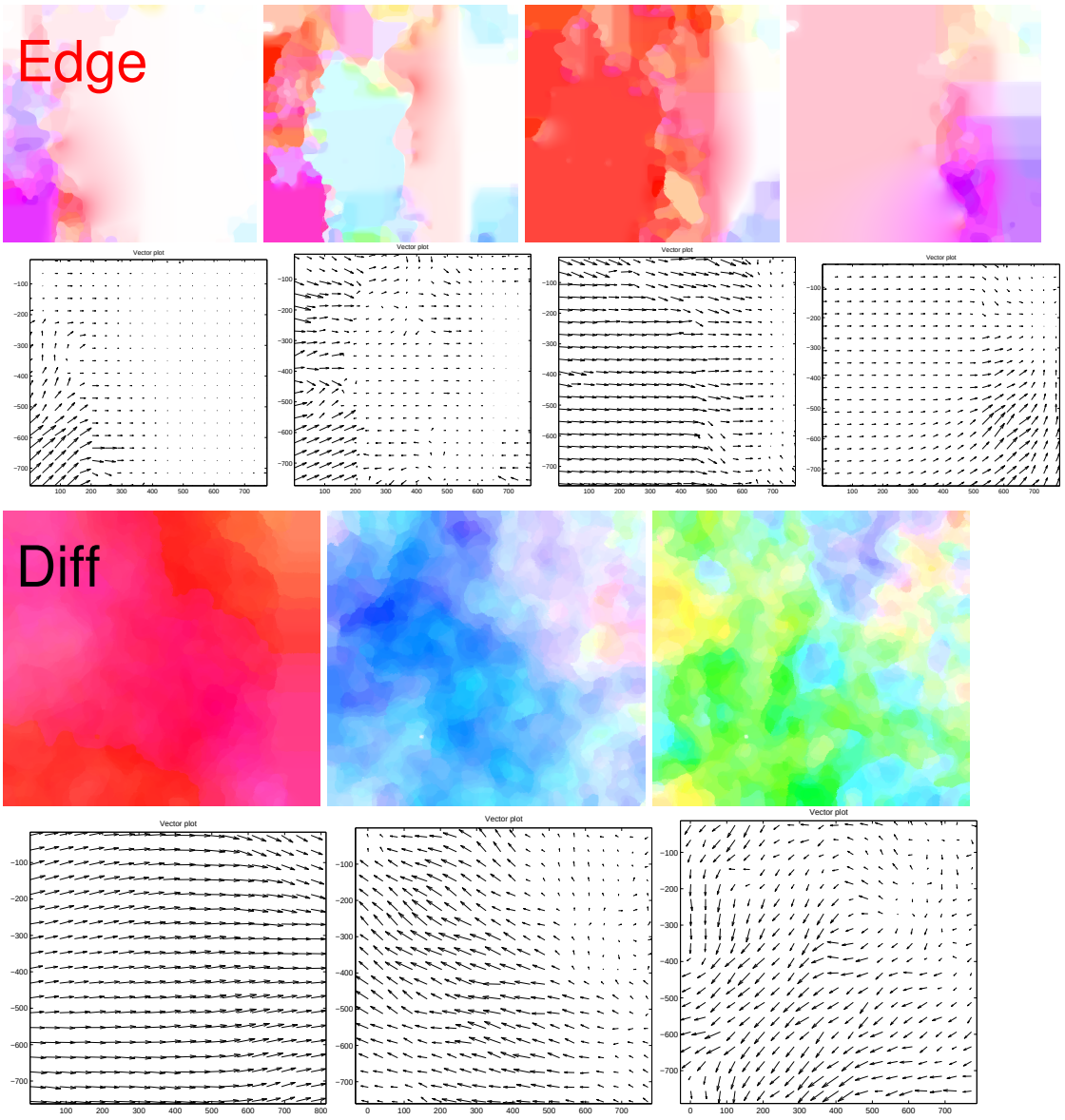



Figure 5.22.: Second part of optical flow implementations on the MR17Z2P2 preprocessed images as Middlebury and vector plot representation. **Edge**: Edge detected images of the original images as Middlebury with the corresponding vector plot in the second row. **Diff**: The image difference between every image and their neighbor as Middlebury with the corresponding vector plot in the fourth row.

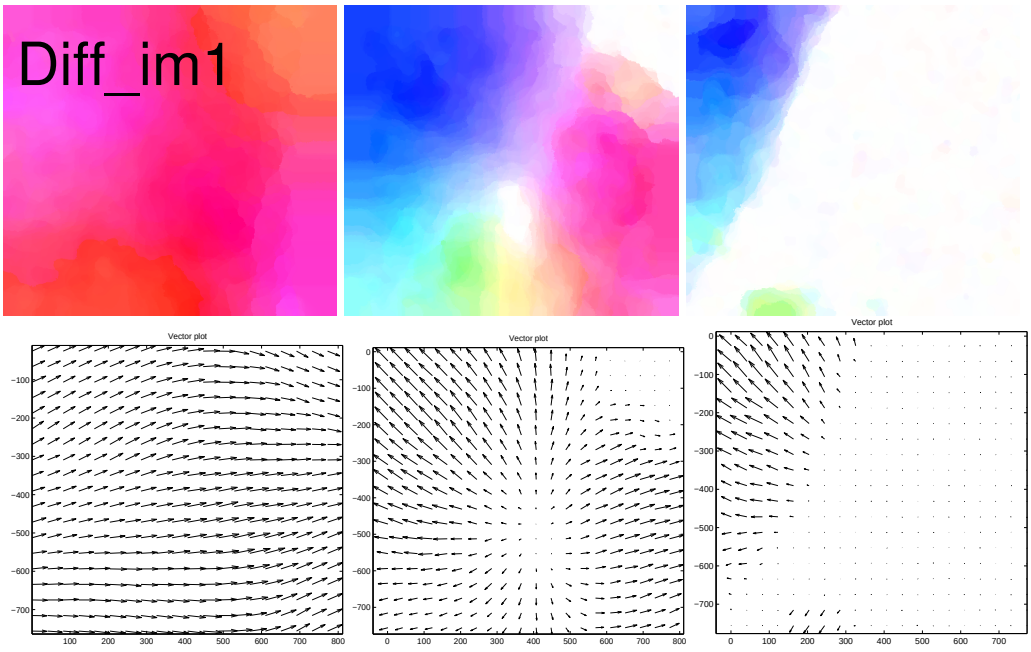



Figure 5.23.: Third part of optical flow implementations on the MR17Z2P2 preprocessed images as Middlebury and vector plot representation. **Diff_im**: The image difference between the every image and the last one as Middlebury with vector plot.

### 5.6.3. MR17Z2P2 Intensity profile

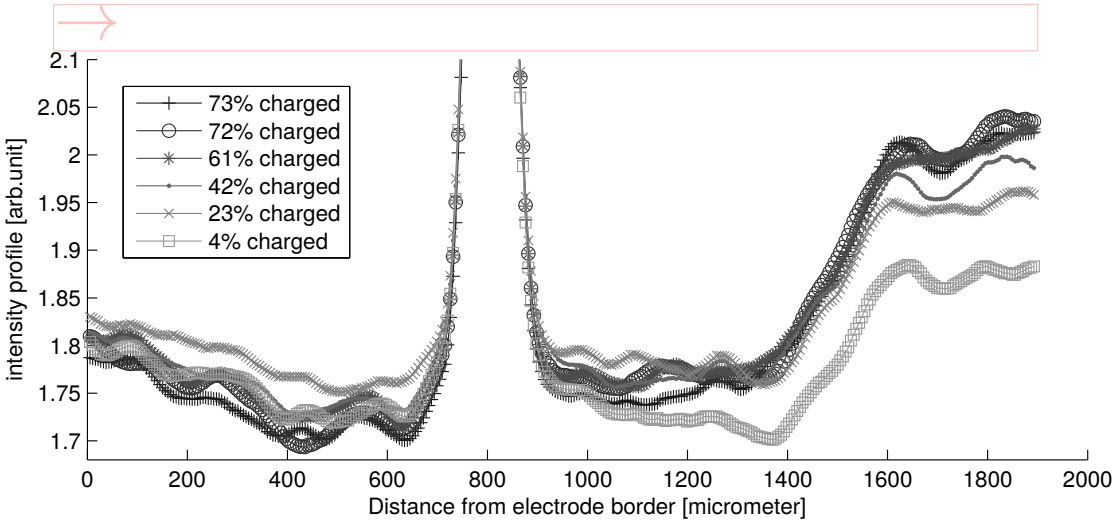



Figure 5.24.: Profile data for MR17Z2P2. The high peak in the middle of the plot signifies part of the electrode cell and, for that reason, is not interesting to our analysis. The same is true for the area left of the peak, which signifies the reference area. Any changes in this area are caused by lighting fluctuations and noise. The actual battery material can be seen on the right side of the peak.

### 5.6.4. MR17Z2P2 Video visualization

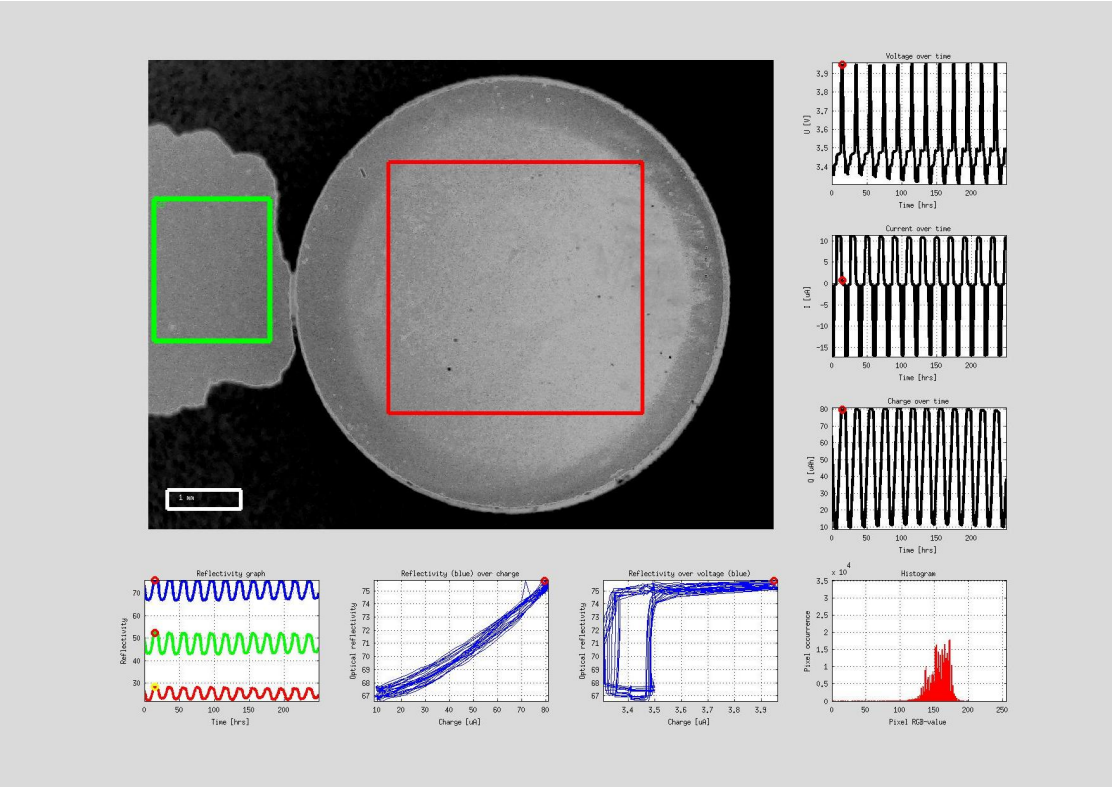



Figure 5.25.: Video frame MR17Z2P2 data set . Grey scaled test cell images during charging and discharging process with the big red rectangle which shows the area of interest and smaller green rectangle which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

### 5.6.5. MR15Z1P2 Video frame

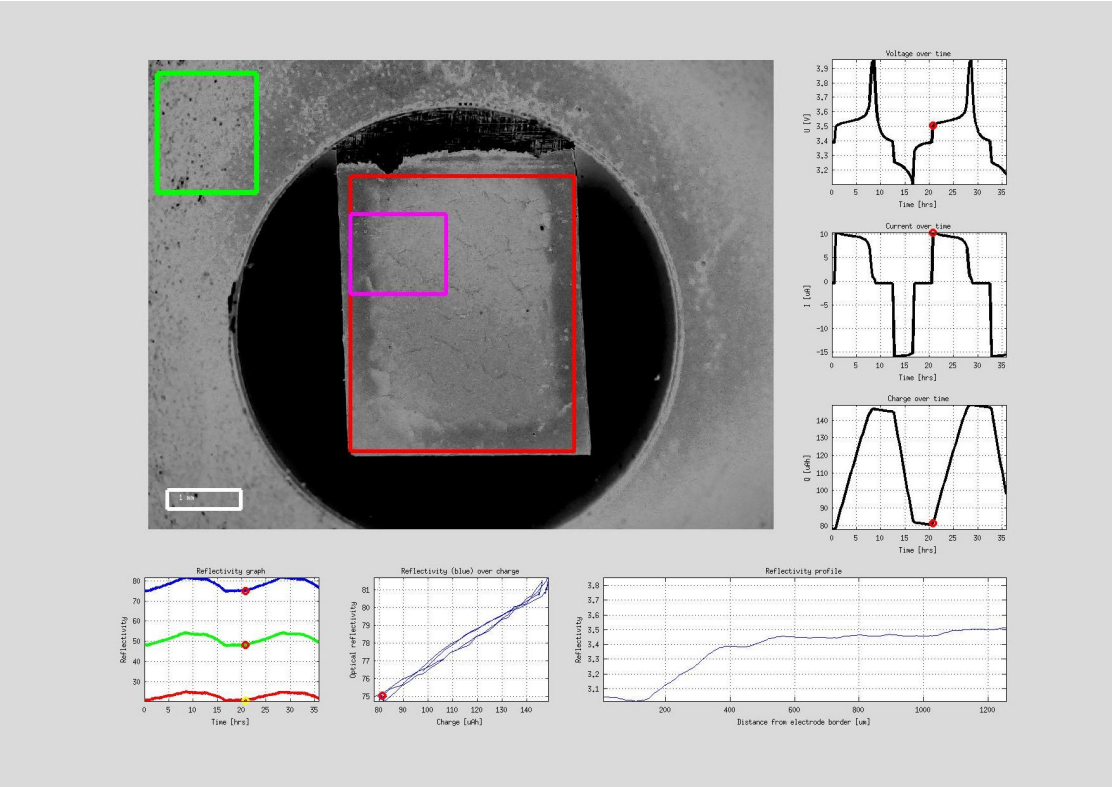



Figure 5.26.: Video frame MR15Z1P2 data set . Test cell Grey scaled images during charging and discharging process with the big red rectangle which shows the area of interest and small pink rectangle which shows the section profile and the small green rectangle which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

### 5.6.6. MR17Z3P1 Video frame

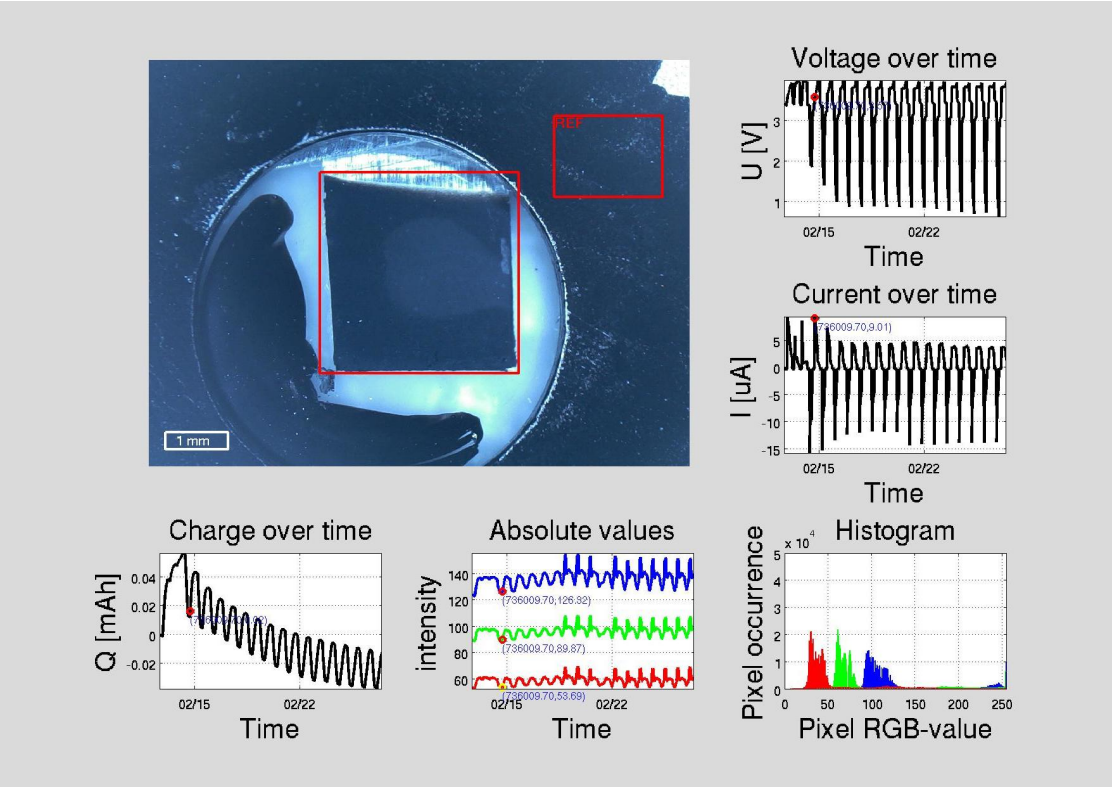



Figure 5.27.: Video frame MR17Z3P1 data set . Test cell images during charging and discharging process with the big red rectangle which shows the area of interest and smaller red rectangle which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

### 5.6.7. MR17Z2P2 Video frame

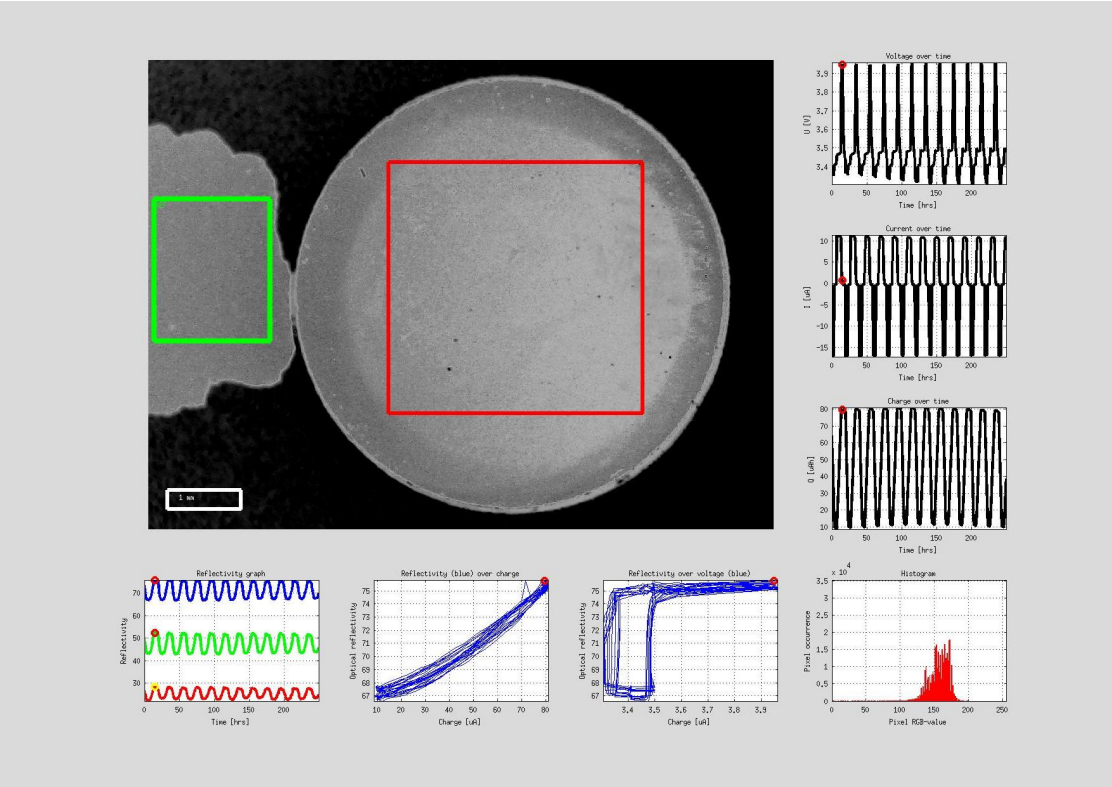



Figure 5.28.: Video frame MR17Z2P2 data set . Grey scaled test cell images during charging and discharging process with the big red rectangle which shows the area of interest and smaller green rectangle which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

### 5.6.8. MR13Z2P2 Video frame

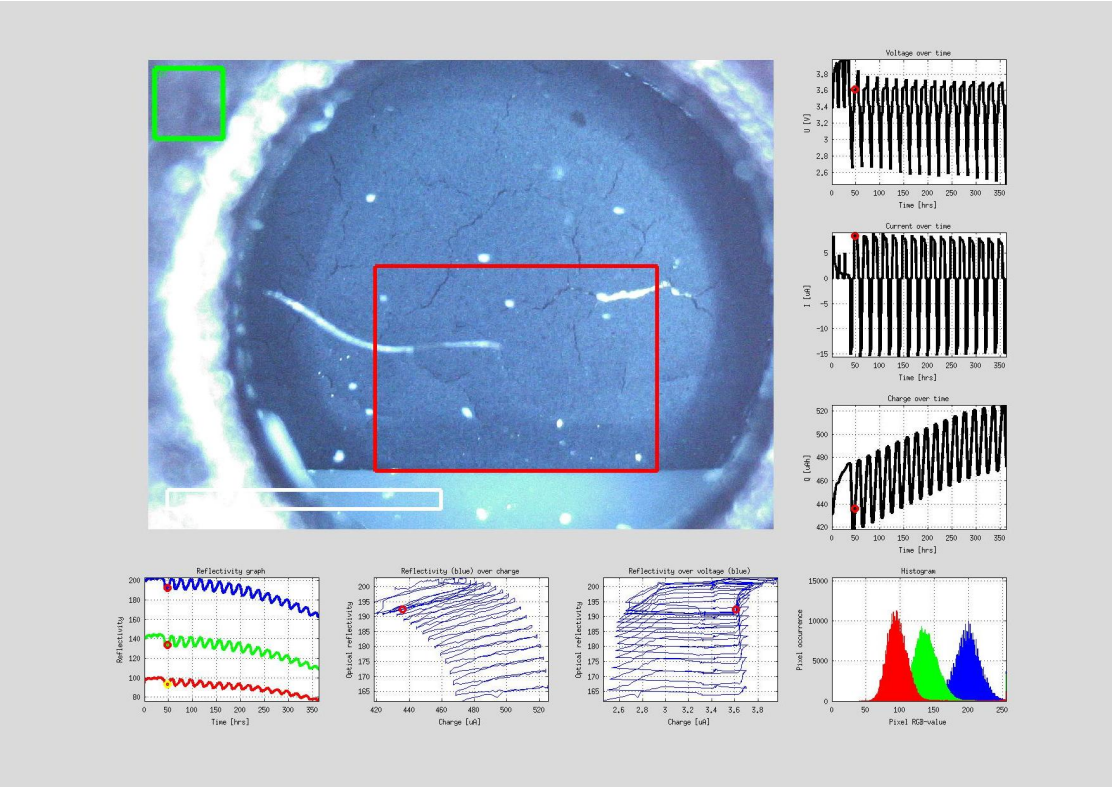



Figure 5.29.: Video frame MR13Z2P2 data set . Test cell images during charging and discharging process with the big red rectangle which shows the area of interest and small green square which shows the area of reference. The graphs present the electrical and optical parameters of test cell images over the period of time.

## 5.7. Scientific analysis

As a result of mass data analysis scientific analysis formulate the comparison between different optical flow implementations to develop testable predictions and general theories. The purpose of optical flow comparison is to identify the novel method that provides accurate information to visualize the optical changes in the battery test cell images during the charging and discharging process. This section with considering all of the constraints compares three optical flow methods which were introduced and implemented on the battery test cell images. It also justify the logic that one of the implementation's results is more effective than the others. The final chosen method can improve further in order to get more accurate results.

**Lucas Kanade** method is introduced in the theory of optical flow. The Lucas Kanade method is purely local method, therefore it cannot provide flow information in the interior of uniform regions of the test cell images. There is no implementation of Lucas Kanade in this thesis but it is proposed for the further development as one of the optical flow methods.

**Horn Schunck** method finds the optical flow by estimating the direction and speed of a moving object from one image to another therefore, it perfectly match the purpose of determining and observing the optical changes from one battery image to another. This method reduces noise in flow because its algorithm assumes smoothness in the flow over the whole image. Its algorithm is more sensitive to noise than Lucas Kanade method but it results in a high density of flow vectors. After implementation of Horn Schunck on different test images and preprocessed battery test cell images, the optical change is observed but still it doesn't provide accurate information to visualize the spatial and temporal distribution of the effect through the images.

**Middlebury** method basically originates from HS method but estimate the flow fields more accurately by optimization methods and modern implementation practice. Its estimation is based on HS method together with Classic+nl with different parameters such as lambda. By increasing the lambda parameter the flow fields will be smoother compare to HS method alone. Implementing Middlebury on the preprocessed battery test cell images results in vector plots and color maps that can be optimized by changing the mentioned parameters and is flexible for further developments.

# 6. Conclusion

This chapter finalize the thesis by introducing the assumptions and summary of discussions and at the end it provides possible ideas to improve based on the current obtained results for the future development of optical flow methodologies. **Investigation of optical effects in lithium batteries with image processing and optical flow** by developing Matlab program modules for image processing, data analysis and visualization of these images data attained successful results, which is included in their related chapters.

## 6.1. Summary

Common battery monitoring concepts use electrical measurements only and estimate the battery state of charge based on a calculated battery model. These methods based on electrical parameters have significant disadvantages therefore, direct observation of the chemical and physical battery state is done with electrical and optical measurement setup together with control softwares in order to capture the battery test cell images during the charging and discharging process. In the previous research of BATSEN group at HAW an optical effect was discovered in the battery test cell images. For lithium iron phosphate cathodes the change of reflection intensity is observed and reproduced. The observation were done with microscope cameras and specialized test cells. This bachelor thesis project continue the research by visualizing the spatial and temporal distribution of the effect during the charging and discharging process of the battery test cells images with the help of the image processing techniques and optical flow methods which were implemented on the preprocessed battery test cell images. As a result, it produces vector plots of optical flow lines of affected area together with their corresponding color map. The visualization is also performed as video with significant diagrams ready for scientific analysis. The plots represent the direction of the intensity change of the battery test cell images. The results are evaluated and compared on several data series. These results can present an accurate estimation and evaluation of the state of battery parameters such as state of charge (SoC) and battery aging. Various preprocessing of the battery test cell images are implemented in Matlab scripts. Gray scaling and Noise reduction (Wiener filter), Image difference with the adjacent images, Binarization, Edge detection

are the main preprocessing steps. After the preprocessing steps, Horn Schunck method has been developed for calculating the optical flow from a sequence of test cell images. It is basically established on the consideration that the flow velocity has two components and rate of change of image brightness will produce only one constraint. The flow smoothness is introduced as the second constraint and an iterative method is developed to solve the resulting equation. The various aspects of the optical flow approaches from the literature, including theory implementation details such as Horn Schunck method and Middlebury optical flow benchmark is studied. Experiment results of each image processing technique are visualized graphically for evaluating and analyzing the series of the results.

## 6.2. Guidance for further developments

The result of this work will be developed further by the BATSEN group at HAW Hamburg. Preprocessing techniques which were done on the test cell images before the optical flow implementation can be improved by considering the brightness and smoothness constraint. For example noise reduction(wiener filter) can be developed further by considering other values and parameters in order to receive less blurred images and at the same time not losing so much information. The other useful preprocessing method is binarization which differentiates the intensity changes in the images and can be used as a concrete preprocessing step before implementing the different optical flow methods therefore, the main purpose of the preprocessing steps which is making the changes visible has to be always considered before optical flow implementation. In the optical flow implementation, Middlebury pattern can be modified by changing the parameters such as Sigma and Lambda for smoother flow fields. This project can be developed further by creating novel methods which are derived from HS methodology and considering the optical constraints of the battery test cell images.

# Bibliography


[1] BALA, Anilet ; HATI, Chiranjeeb ; PUNITH, CH: Image denoising method using curvelet transform and wiener filter. In: *International Journal of Advanced Research in Electrical, Electronics* (2014)

[2] BARRON, John L. ; FLEET, David J. ; BEAUCHEMIN, Steven S.: Performance of optical flow techniques. In: *International journal of computer vision* 12 (1994), Nr. 1, S. 43–77

[3] BEAUCHEMIN, Steven S. ; BARRON, John L.: The computation of optical flow. In: *ACM computing surveys (CSUR)* 27 (1995), Nr. 3, S. 433–466

[4] BOUGUET, Jean-Yves: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. In: *Intel Corporation* 5 (2001), Nr. 1-10, S. 4

[5] BUADES, Antoni ; COLL, Bartomeu ; MOREL, J-M: A non-local algorithm for image denoising. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* Bd. 2 IEEE, 2005, S. 60–65

[6] BUADES, Antoni ; COLL, Bartomeu ; MOREL, Jean-Michel: A review of image denoising algorithms, with a new one. In: *Multiscale Modeling & Simulation* 4 (2005), Nr. 2, S. 490–530

[7] EL-SAYED, M.A.: *Algorithms for Image Thresholding*. Lap Lambert Academic Publishing GmbH KG, 2012. – ISBN 9783659128004

[8] GREEN, Bill: Canny edge detection tutorial. In: *Retrieved: March* 6 (2002), S. 2005

[9] HORN, Berthold K. ; SCHUNCK, Brian G.: Determining optical flow. In: *Artificial intelligence* 17 (1981), Nr. 1-3, S. 185–203

[10] KOMORKIEWICZ, Mateusz ; KRYJAK, Tomasz ; GORGON, Marek: Efficient hardware implementation of the Horn-Schunck algorithm for high-resolution real-time dense optical flow sensor. In: *Sensors* 14 (2014), Nr. 2, S. 2860–2891

[11] LEE, Hun ; YANILMAZ, Meltem ; TOPRAKCI, Ozan ; FU, Kun ; ZHANG, Xiangwu: A review of recent developments in membrane separators for rechargeable lithium-ion batteries. In: *Energy & Environmental Science* 7 (2014), Nr. 12, S. 3857–3886

[12] LEE, Jong-Sen: Digital image enhancement and noise filtering by use of local statistics. In: *IEEE transactions on pattern analysis and machine intelligence* (1980), Nr. 2, S. 165–168

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

[14] LUKAC, R. ; PLATANIOTIS, K.N.: *Color Image Processing: Methods and Applications*. CRC Press, 2006 (Image Processing Series). – ISBN 9781420009781

[15] MARR, D. ; HILDRETH, E.C.: *Theory of Edge Detection*. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1979 (AI memo)

[16] MATHWORKS: *Thresholding & Binarization*. `https://www.mathworks.com/discovery/image-thresholding.html`. Version: 12.04.2017

[17] PADHI, AK ; NANJUNDASWAMY, KS ; GOODENOUGH, JB: LiFePO4: a novel cathode material for rechargeable batteries. In: *Electrochemical Society Meeting Abstracts* Bd. 96, 1996, S. 73

[18] PALLIYAGURUGE, Don Mithila M.: *Image processing for investigation of effects in Lithium battery electrodes*

[19] PRODUCTS: *ECC-Opto-Std*. `https://el-cell.com/products/test-cells/optical-test-cells/ecc-opto-std`. Version: 12.04.2017

[20] RAUDIES, F.: Optic flow. 8 (2013), Nr. 7, S. 30724. – revision 149632

[21] ROSCHER, Valentin: *Personal communication*. 2017

[22] ROSCHER, Valentin ; RIEMSCHNEIDER, Karl-Ragmar: In-situ Electrode Observation as an Optical Sensing Method for Battery State of Charge. In: *Sensors Applications Symposium (SAS), 2017 IEEE* IEEE, 2017, S. 1–6

[23] ROYDEN, Constance S. ; MOORE, Kathleen D.: Use of speed cues in the detection of moving objects by moving observers. In: *Vision research* 59 (2012), S. 17–24

[24] SEZGIN, Mehmet u. a.: Survey over image thresholding techniques and quantitative performance evaluation. In: *Journal of Electronic imaging* 13 (2004), Nr. 1, S. 146–168

[25] SONKA, Milan ; HLAVAC, Vaclav ; BOYLE, Roger: Image pre-processing. In: *Image Processing, Analysis and Machine Vision*. Springer, 1993, S. 56–111

[26] SUN, Deqing ; ROTH, Stefan ; BLACK, Michael J.: Secrets of optical flow estimation and their principles. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* IEEE, 2010, S. 2432–2439

[27] SUN, Deqing ; ROTH, Stefan ; BLACK, Michael J.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. In: *International Journal of Computer Vision* 106 (2014), Nr. 2, S. 115–137

# Appendices

# A. Assignment description

Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

Hochschule für Angewandte Wissenschaften Hamburg
Department Informations- und Elektrotechnik
Prof. Dr.-Ing. Karl-Ragmar Riemschneider

24. Januar 2017

# Bachelor thesis Ali Emami

# Investigation of optical effects in lithium batteries with image processing and the optical flow method

## Motivation

In electric 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 significant 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 charging
- long term changes of cell parameters due to ageing

For that reason the battery research group at HAW Hamburg do research for direct observation of the chemical and physical battery state. In previous work an optical effect was discovered, which is promising for that purpose. For lithium iron phosphate cathodes the change of 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.

## Assignment

The main objective of the thesis is to visualize the spatial and temporal distribution of the effect during the charge and discharge processes. In particular, the method of optical flow should be test in the context before.
Therefore Mr. Ali Emami has to develop Matlab program modules for image processing, data evaluation and visualization of these image data. The platform is a conventional PC with Linux.
Major work packages of the thesis are:

- **Introduction and Basics**
  - Literature analysis and introductory explanations:
  - Problem analysis of battery state estimation
  - State of the research in optical sensing methods for electrochemistry
  - State of the research in image processing with optical flow methods - more detailed comparision
  - Participation in measurement experiments and setup preparation
- **Preprocessing and Compensation of Measurement Problems**
  - Noise cancellation / minimization by stacking of images including data reduction
  - Optional reduction of image shifts by correlation calculation
  - Separation of areas of interest from non active areas

  - Exclusion of areas not affected by charge-discharge effects (e.g. cracks, shadows, surrounding area, dust, (camera) artefacts) by masking or default/limit value setting
  - Optional: Elimination of misalignment in the images (rotation of camera position, non rectangular projection)
  - Optional: Data reduction by reducing full images to relevant image areas

- **Implementation for optical flow determination for image data series / videos**
  - Comparison and selection of optical flow methods
  - Implementation of selected method(s), including performance tests
  - Parameter matching of the method(s) of images (p.e. size of subareas for flow determination, thresholds and average/filtering in time)
  - Visualization of raw images changes parallel to Matlab-quiver / Multicolor diagrams of optical flow
  - Evaluation comparison of results on several data series
  - Optional: Comparison with 'simple' cartesian or polar gradients (X-Direction, Y-Direction) for suitable electrode geometries
  - Optional: Separation of results for camera color channels

- **Effect analysis along the optical flow lines**
  - Analysis of temporal shift of intensity along single gradient-directions and optical flow lines
  - Collective correlation of intensity changes along the optical flow lines of affected areas
  - Visualisation as video and diagrams

- **Functional evaluation and use for mass data analysis**
  - Production of movies for few experiments for visualisation
  - Planning of structured evaluation procedures, e.g. in batch jobs
  - Preparing significant diagrams ready for scientific analysis
  - Software and data handling process documentation

- **Conclusion**
  - Subsumption of the results of the mass data analysis
  - Critical discussion of results
  - Summarize the achieved state and important facts, incl. proposals for future work

# B. Matlab Code

# B.1. config.m

config.m contains the coordinates of the evaluated sections, reference and profiles together with the electrical data.

```
% Version 1.2 2016-10-18 VR - AE added shift_from_UTC, added coordinates of
% the different sections of the image
function param = config()

test_no='MR13Z2P2';
param.test_no=test_no;

%%Coordinate of the evaluated section
param.xposl = 298;
param.xposr = 1365;
param.yposu = 464;
param.yposd = 900;

%%
%%coordinates of the reference
param.ref_xposl = 1508;
param.ref_xposr = 1570;
param.ref_yposu = 824;
param.ref_yposd = 890;
%%
% Number of sections for profile plots
param.profiles=1;
%%
%%Coordinate of the section_A
param.profileA_xposl = 652;
param.profileA_xposr = 1022;
param.profileA_yposu = 494;
param.profileA_yposd = 904;
%%
%%Coordinate of the section_B
param.profileB_xposl = 100;
param.profileB_xposr = 200;
param.profileB_yposu = 100;
param.profileB_yposd = 200;
%%
%%Coordinate of the section_C
param.profileC_xposl = 652;
param.profileC_xposr = 1022;
param.profileC_yposu = 494;
param.profileC_yposd = 904;
%%
% One mm in pixels (calculate from hole diameter!)
param.onemm = 216;
%%
% 1 for winter, 2 for summer, 8 for China
param.shift_from_UTC = 1;
%%
% Electrical data. Use CurrentIntegrator to generate chargefile!
param.voltageFile = '../Data/20140927_1530_zelle_2_spannung.txt';
param.currentFile = '../Data/20140927_1530_zelle_2_strom.txt';
param.chargeFile = '../Data/20140927_1530_zelle_2_ladung.txt';
%%



param.profile_data = '../Output/Data/',test_no,'_profile';
param.imageSource = '../Images/';
param.imageDest = '../Output/Images/';
param.dataDest = '../Output/Data/';
param.plotDest = '../Output/Plots/',test_no;    % Contains part of the
    filename!

param.tempVolt = strcat('../Output/Data/',test_no,'_Voltdata.txt');
param.tempCurrent = strcat('../Output/Data/',test_no,'_Currentdata.txt');
param.tempCharge = strcat('../Output/Data/',test_no,'_Chargedata.txt');

param.ImageRGB_data = strcat('../Output/Data/',test_no,'_RGBdata');
param.imageDest = strcat('../Output/Images/');
param.dataDest = strcat('../Output/Data/',test_no);
param.plotDest = strcat('../Output/Plots/',test_no);
param.imageTimeStamps = strcat('../Output/Data/',test_no,'
    _imageTimeStamps.txt');
param.videoframes = strcat('../Output/Images/Videoframes/',test_no);
```

# B.2. m01_imageRead_v02.m

m01_imageRead_v02.m reads images. It is used to preprocess the images. It uses the information from the config.m file and then reads the raw image data as well as the electrical measurement data.

```
%% v1.8 2016-11-23 DMMP, VR, AE
% v1.7 2016-10-04 DMMP, VR,WSQ, VR, AE
%  v1.7     Added spatial distribution analysis
% v1.6:     Added shift_from_UTC from config.m file
%  v1.5:    Removed most Plotting functions, only kept the basic plots
%  v1.4:
%  v1.3:    Removed Plot function, renamed
%  v1.2:    Merged with Plot function from rgbvoltcurrentchargeplotter

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

%%


test = 0;                           % test = 1 only evaluates first
    10 pictures
parser = config();                  % this codes links to config.m
    file in the working directory.
profiling = parser.profiles;        % profiling = 1 evaluates
    profiles and image processing. Can be set here or in config file.

%% Input/Output files


%% allocate memory for array %%number of parts are usually one
xpos      = zeros(1);
ypos      = zeros(1);
xsize     = zeros(1);
ysize     = zeros(1);
xpos_ref  = zeros(1);
ypos_ref  = zeros(1);
xsize_ref = zeros(1);
ysize_ref = zeros(1);
spa_xpos  = zeros(1);
spa_ypos  = zeros(1);
spa_xsize = zeros(1);
spa_ysize = zeros(1);
spb_xpos  = zeros(1);
spb_ypos  = zeros(1);
spb_xsize = zeros(1);
spb_ysize = zeros(1);
spc_xpos  = zeros(1);
spc_ypos  = zeros(1);
spc_xsize = zeros(1);
spc_ysize = zeros(1);
path      = cell(1);
channels  = cell(7); % RGB timestamps RGB(reference)
noFiles   = zeros(1);

path = parser.imageSource;

% get all .png files in given directory
files = dir(fullfile(char(path), '*.png'));

% pick number of files to be read and num of pixels
noFiles   = size(files, 1);
imname= cell(noFiles, 1);



if test == 1
    noFiles = 10;
end

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



%% get paths and section information


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

    xpos_ref  = parser.ref_xposl;
    ypos_ref  = parser.ref_yposu;
    xsize_ref = parser.ref_xposr - parser.ref_xposl;
    ysize_ref = parser.ref_yposd - parser.ref_yposu;


    noPixels     = xsize * ysize;
    noPixels_ref = xsize_ref * ysize_ref;

    clear invalues;

if profiling == 1;
    spa_xpos= parser.profileA_xposl;
```

```
    spa_ypos= parser.profileA_yposu;
    spa_xsize= parser.profileA_xposr-parser.profileA_xposl;
    spa_ysize= parser.profileA_yposd-parser.profileA_yposu;

    spb_xpos= parser.profileB_xposl;
    spb_ypos= parser.profileB_yposu;
    spb_xsize= parser.profileB_xposr-parser.profileB_xposl;
    spb_ysize= parser.profileB_yposd-parser.profileB_yposu;

    spc_xpos= parser.profileC_xposl;
    spc_ypos= parser.profileC_yposu;
    spc_xsize= parser.profileC_xposr-parser.profileC_xposl;
    spc_ysize= parser.profileC_yposd-parser.profileC_yposu;

    noPixelsProfileA     = spa_xsize * spa_ysize;
    noPixelsProfileB     = spb_xsize * spb_ysize;
    noPixelsProfileC     = spc_xsize * spc_ysize;

    intensity_profile_A = zeros(noFiles, 1);
    intensity_profile_B = zeros(noFiles, 1);
    intensity_profile_C = zeros(noFiles, 1);
end
%% process images
    %  we will get all of the figures from here and then make video out
        of it, produce figures which shows pixels 1200*1600 and make
        video out of it.



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

        % extract sections as defined by config.m
        section      = A(ypos:ypos+ysize-1, xpos:xpos+xsize-1, :);
        section_ref = A(ypos_ref:ypos_ref+ysize_ref-1, xpos_ref:xpos_ref+
            xsize_ref-1, :);


        % 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;
        % sum-up intensities on every channel for reference area
        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;


if profiling ==1;
        %%% Experimental image manipulation %%%

        section_profile_A = A(spa_ypos:spa_ypos+spa_ysize-1, spa_xpos:
            spa_xpos+spa_xsize-1, :);
        section_profile_B = A(spb_ypos:spb_ypos+spb_ysize-1, spb_xpos:
            spb_xpos+spb_xsize-1, :);
        section_profile_C = A(spc_ypos:spc_ypos+spc_ysize-1, spc_xpos:
            spc_xpos+spc_xsize-1, :);

        % Save section to file for later use
        imwrite(section,strcat(parser.imageDest,'Section/section_',
            num2str(n),'.jpg'));


        % Calculate difference between images and save to file
        if n==1
            section_image1=section;
        else
            section_diff=((section-section_old)+10)*10;
            imwrite(section_diff,strcat(parser.imageDest,'Section_diff/
                section_diff_',num2str(n),'.jpg'));
            section_diff_from_image1=((section-section_image1)+10)*10;
            imwrite(section_diff_from_image1,strcat(parser.imageDest,'
                Section_diff_from_image1/section_diff_from_image1_',
                num2str(n),'.jpg'));

        end;


        if n==1
            section_image1=section;
        end;

        section_old=section;        % Save old section for gradient



        % sum-up intensities for Profile A
        for o=1:1:spa_xsize-10
            intensity_profile_A(n,o) = sum(sum(section_profile_A(:, o:o
                +10, 3)))/noPixelsProfileA; %% Three dimension matrix
        end

        % sum-up intensities for Profile B

        for o=1:1:spb_xsize-10
            intensity_profile_B(n,o) = sum(sum(section_profile_B(:, o:o
                +10, 3)))/noPixelsProfileB; %% Three dimension matrix
        end
        % sum-up intensities for Profile C, evaluated vertically
        for o=1:1:spc_xsize-10
            intensity_profile_C(n,o) = sum(sum(section_profile_C(o:o+10,
                :, 3)))/noPixelsProfileC; %% Three dimension matrix
        end
        % Clear sections


end
        %%% End experimental image manipulation %%%



        clear A MAP ALPHA;
        disp(['File: ', num2str(n), '/', num2str(noFiles)]);

        clear section section_ref section_diff;

        %%% plot the pictures , the code didn't plot here, therefore the
        %%% matlab command window is used !
%         %% pic 1
%         x=[10,20,30,40];
%         y=[100,25,30,75];
%         plot(x,y);
%         %pic 2
%         x=[10,20,30,40];
%         z=[100,30,50,75];
%         plot(x,y);
%         %pic3
%         x=[10,20,30,40];
%         P=[100,50,50,75];
%         plot(x,P);
%         %pic4
%         x=[10,20,30,40];
%         Q=[100,75,75,75];
%         plot(X,Q);
%
    end


% concatenate parts, probably not needed anymore due to removal of parts.

channels{1} = red(:,1);
channels{2} = green(:,1);
channels{3} = blue(:,1);
channels{4} = dates;
channels{5} = red(:,2);
channels{6} = green(:,2);
channels{7} = blue(:,2);
clear n red green blue dates files noPixels noPixels_ref;

% concatenate parts, probably not needed anymore due to removal of parts.
red       = vertcat(channels{1, :});
green     = vertcat(channels{2, :});
blue      = vertcat(channels{3, :});
dates     = vertcat(channels{4, :});
red_ref   = vertcat(channels{5, :});
green_ref = vertcat(channels{6, :});
blue_ref  = vertcat(channels{7, :});
clear channels m;


%% apply correction to improve the intensity measurement and remove
      effects such as LED ageing and temperature changes
red_c   = (red   ./ red_ref)   * mean(red_ref);
green_c = (green ./ green_ref) * mean(green_ref);
blue_c  = (blue  ./ blue_ref)  * mean(blue_ref);


%% export data to a text and a mat file
exportfile = strcat(parser.ImageRGB_data);
udates = round(864e5 * (dates - datenum('1970', 'yyyy')))./1000-parser.
      shift_from_UTC*3600;% Daylight Saving time: -3600, Summer time:
      -7200
exportMatrix = [udates red green blue red_c green_c blue_c];
dlmwrite(exportfile, sprintf('#timestamp\tred (raw)\tgreen (raw)\tblue (
      raw)\tred (compensated)\tgreen (compensated)\tblue (compensated)')
      , 'delimiter', '');
dlmwrite(exportfile, exportMatrix, '-append', 'delimiter', '\t', '
      precision', 15);
save(strcat(parser.ImageRGB_data,'.mat'),'exportMatrix','imname');
clear exportMatrix;

%% Export profile data - updated on 2016.10.04 AE

if profiling ==1;

    %Profile_A
    exportMatrix = [intensity_profile_A];
    save(strcat(parser.profile_data,'_A.mat'),'exportMatrix','imname');
```

```
    clear exportMatrix;
    %
    %Profile_B
    exportMatrix = [intensity_profile_B];
    save(strcat(parser.profile_data,'_B.mat'),'exportMatrix','imname');
    clear exportMatrix;

    %Profile_C
    exportMatrix = [intensity_profile_C];
    save(strcat(parser.profile_data,'_C.mat'),'exportMatrix','imname');
    clear exportMatrix;

end
%%
% get data as a mat file
A = load(strcat(parser.ImageRGB_data,'.mat'));
filenames = A.imname(:,1);
%%
% get the image time stamps to a text file
[status dates] = system(['sed "1{/^#/d;};s/[\t].*//g" ' parser.
    ImageRGB_data]);
disp(dates);
%%
% to do: compare udates, dates for high difference to prevent !!!!!??????
% TimestampMatcher misbehavior
%%
% save image timestamps in a temporary text file
fid = fopen(parser.imageTimeStamps, 'wt');
fprintf(fid, '%.0f\n', udates);
%%
%
system(['cat ' parser.voltageFile '|grep -v "#"|./TimestampMatcher -f '
    parser.imageTimeStamps ' >' parser.tempVolt]);
system(['cat ' parser.currentFile '|grep -v "#"|./TimestampMatcher -f '
    parser.imageTimeStamps ' >' parser.tempCurrent]);
system(['cat ' parser.chargeFile '|grep -v "#"|./TimestampMatcher -f '
    parser.imageTimeStamps ' >' parser.tempCharge]);
%
% TimestampMatcher output is faulty if it cannot find a match within
    ~1000
% seconds. First lines or complete output will be garbled. Also check
% plausibility of data for DST/time zone mismatch
%
% To match only correct formatting (I could not get \tab and -? to work,
    so two greps):
% system(['cat ' parser.voltageFile '|grep -E "^[0-9]{10}\.[0-9]{3}"|grep
    -E "[0-9]{1}\.[0-9]{6}E[+-][0-9]{2}$"|./TimestampMatcher -f ./
    imageTimeStamps.txt >' parser.tempVolt]);
% system(['cat ' parser.currentFile '|grep -E "^[0-9]{10}\.[0-9]{3}"|grep
    -E "[0-9]{1}\.[0-9]{6}E[+-][0-9]{2}$"|./TimestampMatcher -f ./
    imageTimeStamps.txt >' parser.tempCurrent]);
% system(['cat ' parser.chargeFile '|grep -E "^[0-9]{10}\.[0-9]{3}"|grep
    -E "[0-9]{1}\.[0-9]{6}E[+-][0-9]{2}$"|./TimestampMatcher -f ./
    imageTimeStamps.txt >' parser.tempCharge]);
%


voltMatrix    = dlmread(parser.tempVolt);
currentMatrix = dlmread(parser.tempCurrent);
chargeMatrix  = dlmread(parser.tempCharge);

%% Plot data
% Define variables for plotting
unixTime = A.exportMatrix(:,1);
imageTime = (unixTime-A.exportMatrix(1,1))/3600;    % Time from start in
    hours. If using imageDate, uncomment datetick commands and adjust
    label!
date = chargeMatrix(:,1);                           % Date/time
chargeData = chargeMatrix(:,2)/3600*1000*1000;      % Charge in uAh
voltData = voltMatrix(:,2);                         % Voltage in V
currentData = currentMatrix(:,2)*1000*1000;         % Current in uA
color_red_unc = A.exportMatrix(:,2);
color_green_unc = A.exportMatrix(:,3);
color_blue_unc = A.exportMatrix(:,4);
color_red_cor = A.exportMatrix(:,5);
color_green_cor = A.exportMatrix(:,6);
color_blue_cor = A.exportMatrix(:,7);
color_all_cor   = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
    exportMatrix(:,7));

%% Charge and Intensity over time for red, green, blue
cit = figure(1);
set(cit,'Units','centimeters');
pos = get(cit,'Position');
set(cit,'PaperPositionMode','auto');
set(gca,'linewidth',1.5);
set(gca,'GridLineStyle','-');
set(cit, 'Position',[0 0 29.7 21]);
set(cit,'PaperOrientation','landscape');
%%
% Charge and Intensity over time for red
subplot(3,1,1);
[ax1,p1,p2] = plotyy(imageTime,chargeData,imageTime,color_red_cor,'plot')
    ;
%datetick(ax1(1),'keeplimits');datetick(ax1(2),'keeplimits');
set(p1,'color','K');set(p2,'color','R')
xlabel('Time [hrs]','FontSize',12);
ylabel(ax1(1),'Charge [uAh]','FontSize',12);
ylabel(ax1(2),'Intensity - Red','FontSize',12);
legend('charge','Intensity - Red');
title('Charge and Intensity - Red over Time','FontSize',12);
grid on;

% Charge and intensity over time for green
subplot(3,1,2);
[ax2,p3,p4] = plotyy(imageTime,chargeData,imageTime,color_green_cor,'plot
    ');
%datetick(ax2(1),'keeplimits');datetick(ax2(2),'keeplimits');
set(p3,'color','K');set(p4,'color','G')
xlabel('Time [hrs]','FontSize',12);
ylabel(ax2(1),'Charge [uAh]','FontSize',12);
ylabel(ax2(2),'Intensity - Green','FontSize',12);
legend('charge','Intensity - Green');
title('Charge and Intensity - Green over Time','FontSize',12);
grid on;

% Charge and intensity over time for blue
subplot(3,1,3);
[ax3,p5,p6] = plotyy(imageTime,chargeData,imageTime,color_blue_cor,'plot
    ');
%datetick(ax3(1),'keeplimits');datetick(ax3(2),'keeplimits');
set(p5,'color','K');set(p6,'color','B')
xlabel('Time [hrs]','FontSize',12);
ylabel(ax3(1),'Charge [uAh]','FontSize',12);
ylabel(ax3(2),'Intensity - Blue','FontSize',12);
legend('charge','Intensity - Blue');
title('Charge and Intensity - Blue over Time','FontSize',12);
linkaxes([ax1,ax2,ax3],'x')
grid on;

%% Voltage, Current, Charge, Intensity over time
vcci = figure(2);
set(vcci,'Units','centimeters');
pos = get(vcci,'Position');
set(vcci,'PaperPositionMode','auto');
grid on
set(vcci,'PaperOrientation','landscape');

ax1 = subplot(4,1,1);
plot(imageTime,voltData,'k');
xlabel('Time [hrs]');
ylabel('Voltage [V]');
title('Voltage over Time');
axis tight;
grid on;

ax2 = subplot(4,1,2);
plot(imageTime,currentData,'k');
xlabel('');
ylabel('Current [uA]');
title('Current over Time');
axis tight;
grid on;

ax3 = subplot(4,1,3);
plot(imageTime,chargeData,'k');
xlabel('');
ylabel('Charge [uAh]');
title('Charge over Time');
axis tight;
grid on;

ax4 = subplot(4,1,4);
plot(imageTime, color_red_cor, 'r', imageTime, color_green_cor, 'g',
    imageTime, color_blue_cor, 'b','LineWidth',2);
xlabel('');
ylabel('intensity[arb.unit]');
title('Absolute values of Intensity, Compensated');
axis tight;
grid on;
grid on;

linkaxes([ax1,ax2,ax3, ax4],'x')
%%

if profiling ==1;
    % Gradient -- Intensity_profile per pixel over time(charging,
        discharging period)
    %profile_interval=[1 9 17 129];
    profile_interval=[int8(noFiles*0.1) int8(noFiles*0.2) int8(noFiles
        *0.3) int8(noFiles*0.4)];
    %Profile_A of the image1, image2, image3
    profileA = figure(3);
    set(profileA,'Units','centimeters');
    pos = get(profileA,'Position');
    set(profileA,'PaperPositionMode','auto');
    grid on
    set(profileA,'PaperOrientation','landscape');
    profile_A_min=min(min(intensity_profile_A(:,:)));
    profile_A_max=max(max(intensity_profile_A(:,:)));

    ax1 = subplot(4,1,1);
    plot(intensity_profile_A(profile_interval(1),:));
    xlabel('Pixel Position');
```

```
ylabel('Intensity[arb.unit]');
ylim([profile_A_min profile_A_max]);
title('Image-1-Profile-A');
grid on;

ax2 = subplot(4,1,2);
plot(intensity_profile_A(profile_interval(2),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_A_min profile_A_max]);
title('Image-2-Profile-A');
grid on;

ax3 = subplot(4,1,3);
plot(intensity_profile_A(profile_interval(3),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_A_min profile_A_max]);
title('Image-3-Profile-A');
axis tight;
grid on;

ax4 = subplot(4,1,4);
plot(intensity_profile_A(profile_interval(4),:));
xlabel('');
ylabel('Intensity[arb.unit');
ylim([profile_A_min profile_A_max]);
title('intensity profile_A over time');
grid on;
linkaxes([ax1,ax2, ax3, ax4],'x');

%% updated on 2016.10.05  AE
%Profile_B of the image1, image2, image3
profileB = figure(4);
set(profileB,'Units','centimeters');
pos = get(profileB,'Position');
set(profileB,'PaperPositionMode','auto');
grid on
set(profileB,'PaperOrientation','landscape');
profile_B_min=min(min(intensity_profile_B(:,:)));
profile_B_max=max(max(intensity_profile_B(:,:)));


ax1 = subplot(4,1,1);
plot(intensity_profile_B(profile_interval(1),:));
xlabel('Pixel Position');
ylabel('Intensity[arb.unit]');
ylim([profile_B_min profile_B_max])
% ylim([min(intensity_profile_B(1,:)) max(intensity_profile_B(1,:) )
    ])
title('Image-1-Profile-B');
grid on;

ax2 = subplot(4,1,2);
plot(intensity_profile_B(profile_interval(2),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_B_min profile_B_max])
% ylim([min(intensity_profile_B(9,:)) max(intensity_profile_B(9,:) )
    ])
title('Image-2-Profile-B');
grid on;

ax3 = subplot(4,1,3);
plot(intensity_profile_B(profile_interval(3),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_B_min profile_B_max])
% ylim([min(intensity_profile_B(,:)) max(intensity_profile_B(17,:) )
    ])
title('Image-3-Profile-B');
grid on;

ax4 = subplot(4,1,4);
plot(intensity_profile_B(profile_interval(4),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_B_min profile_B_max])
% ylim([min(intensity_profile_B(129,:)) max(intensity_profile_B
    (129,:) )])
title('intensity profile_B over time');
grid on;
linkaxes([ax1,ax2, ax3, ax4],'x');

%Profile_C of the image1, image2, image3
profileC = figure(5);
set(profileC,'Units','centimeters');
pos = get(profileC,'Position');
set(profileC,'PaperPositionMode','auto');
grid on
set(profileC,'PaperOrientation','landscape');
profile_C_min=min(min(intensity_profile_C(:,:)));
profile_C_max=max(max(intensity_profile_C(:,:)));

ax1 = subplot(4,1,1);
plot(intensity_profile_C(profile_interval(1),:));
xlabel('Pixel Position');
ylabel('Intensity[arb.unit]');
ylim([profile_C_min profile_C_max]);
title('Image-1-Profile-C');
grid on;

ax2 = subplot(4,1,2);
plot(intensity_profile_C(profile_interval(2),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_C_min profile_C_max]);
title('Image-2-Profile-C');
grid on;

ax3 = subplot(4,1,3);
plot(intensity_profile_C(profile_interval(3),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_C_min profile_C_max]);
title('Image-3-Profile-C');
grid on;

ax4 = subplot(4,1,4);
plot(intensity_profile_C(profile_interval(4),:));
xlabel('');
ylabel('Intensity[arb.unit]');
ylim([profile_C_min profile_C_max]);
title('intensity profile_C over time');
grid on;
grid on;
linkaxes([ax1,ax2, ax3, ax4],'x');
end
```

# B.3. m02_plotCreate.m

m02_plotCreate.m is linked to the config.m file and receives the electrical and optical parameters

and plot their plots during the charging and discharging process of the battery.

```
%% 2016-09-23 DMMP, VR, WSQ, AE
%  v1.5: Add the Plot fuction
%  v1.4: Cleaned up functions, fixed graphs
%  v1.3: Removed Plot function, renamed
%  v1.2:Merged with Plot function from rgbvoltcurrentchargeplotter

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

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

% get data as a mat file
A = load(strcat(parser.ImageRGB_data,'.mat'));
voltMatrix    = dlmread(parser.tempVolt);
currentMatrix = dlmread(parser.tempCurrent);
chargeMatrix  = dlmread(parser.tempCharge);
%delete (parser.tempVolt,parser.tempCurrent,parser.tempCharge);

%% Plot data

unixTime = A.exportMatrix(:,1);
imageTime = (unixTime-A.exportMatrix(1,1))/3600;    % Time from start in
    hours. If using imageDate, uncomment datetick commands and adjust
    label!
date = chargeMatrix(:,1);                           % Date/time
chargeData = chargeMatrix(:,2)/3600*1000*1000;      % Charge in uAh
voltData = voltMatrix(:,2);                         % Voltage in V
currentData = currentMatrix(:,2)*1000*1000;         % Current in uA
color_red_unc = A.exportMatrix(:,2);
color_green_unc = A.exportMatrix(:,3);
color_blue_unc = A.exportMatrix(:,4);
color_red_cor = A.exportMatrix(:,5);
color_green_cor = A.exportMatrix(:,6);
color_blue_cor = A.exportMatrix(:,7);
color_all_cor   = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
    exportMatrix(:,7));


%% Charge and Intensity over time for red, green, blue

cit = figure(1);
set(cit,'Units','centimeters');
pos = get(cit,'Position');
set(cit,'PaperPositionMode','auto');
set(gca,'linewidth',1.5);
set(gca,'GridLineStyle','-');
set(cit, 'Position',[0 0 29.7 21]);
set(cit,'PaperOrientation','landscape');
```

```
% Charge and Intensity over time for red
subplot(3,1,1);
[ax1,p1,p2] = plotyy(imageTime,chargeData,imageTime,color_red_cor,'plot')
    ;
%datetick(ax1(1),'keeplimits');datetick(ax1(2),'keeplimits');
set(p1,'color','K');set(p2,'color','R')
xlabel('Time [hrs]','FontSize',12);
ylabel(ax1(1),'Charge [uAh]','FontSize',12);
ylabel(ax1(2),'Intensity - Red [arb. units]','FontSize',12);
legend('charge','Intensity - Red');
title('Charge and Intensity - Red over Time','FontSize',12);
grid on;

% Charge and intensity over time for green
subplot(3,1,2);
[ax2,p3,p4] = plotyy(imageTime,chargeData,imageTime,color_green_cor,'plot
    ');
%datetick(ax2(1),'keeplimits');datetick(ax2(2),'keeplimits');
set(p3,'color','K');set(p4,'color','G')
xlabel('Time [hrs]','FontSize',12);
ylabel(ax2(1),'Charge [uAh]','FontSize',12);
ylabel(ax2(2),'Intensity - Green [arb. units]','FontSize',12);
legend('charge','Intensity - Green');
title('Charge and Intensity - Green over Time','FontSize',12);
grid on;

% Charge and intensity over time for blue
subplot(3,1,3);
[ax3,p5,p6] = plotyy(imageTime,chargeData,imageTime,color_blue_cor,'plot
    ');
%datetick(ax3(1),'keeplimits');datetick(ax3(2),'keeplimits');
set(p5,'color','K');set(p6,'color','B')
xlabel('Time [hrs]','FontSize',12);
ylabel(ax3(1),'Charge [uAh]','FontSize',12);
ylabel(ax3(2),'Intensity - Blue [arb. units]','FontSize',12);
legend('charge','Intensity - Blue');
title('Charge and Intensity - Blue over Time','FontSize',12);
linkaxes([ax1,ax2,ax3],'x')
grid on;

%% Voltage, Current, Charge, Intensity over time
vcci = figure(2);
set(vcci,'Units','centimeters');
pos = get(vcci,'Position');
set(vcci,'PaperPositionMode','auto');
grid on
set(vcci,'PaperOrientation','landscape');

ax1 = subplot(4,1,1);
plot(imageTime,voltData,'k');
xlabel('Time [hrs]');
ylabel('Voltage [V]');
title('Voltage over Time');
axis tight;
grid on;

ax2 = subplot(4,1,2);
plot(imageTime,currentData,'k');
xlabel('Time [hrs]');
ylabel('Current [uA]');
title('Current over Time');
axis tight;
grid on;

ax3 = subplot(4,1,3);
plot(imageTime,chargeData,'k');
xlabel('Time [hrs]');
ylabel('Charge [uAh]');
title('Charge over Time');
axis tight;
grid on;

ax4 = subplot(4,1,4);
plot(imageTime, color_red_cor, 'r', imageTime, color_green_cor, 'g',
    imageTime, color_blue_cor, 'b','LineWidth',2);
xlabel('Time [hrs]');
ylabel('Intensity [arb. units]');
title('Absolute values of Intensity, Compensated');
axis tight;
grid on;
grid on;

linkaxes([ax1,ax2,ax3, ax4],'x')

%% Intensity Raw Data
ird = figure(3);
set(ird,'Units','centimeters');
pos = get(ird,'Position');
set(ird,'PaperPositionMode','auto');
set(ird,'PaperOrientation','landscape');

ax1 = subplot(3,1,1);
plot(imageTime, color_red_unc, 'r', imageTime, color_green_unc, 'g',
    imageTime, color_blue_unc, 'b','LineWidth',2);
xlabel('Time [hrs]');
ylabel('Intensity [a. u.]');
title('Optical intensity, uncorrected');
axis tight;
grid on;

ax2 = subplot(3,1,2);
plot(imageTime, color_red_cor, 'r', imageTime, color_green_cor, 'g',
    imageTime, color_blue_cor, 'b','LineWidth',2);
xlabel('Time [hrs]');
ylabel('Intensity [a. u.]');
title('Optical intensity, corrected');
axis tight;
grid on;

ax3 = subplot(3,1,3);
plot(imageTime, color_red_cor-color_red_unc, 'r', imageTime,
    color_green_cor-color_green_unc, 'g', imageTime, color_blue_cor-
    color_blue_unc, 'b','LineWidth',2);
xlabel('Time [hrs]');
ylabel('Intensity [a. u.]');
title('Difference between corrected and uncorrected optical intensity');
axis tight;
grid on;

linkaxes([ax1,ax2,ax3],'x')

%% Current und voltage over charge

cvc = figure(4);
set(cvc,'Units','centimeters');
pos = get(cvc,'Position');
set(cvc,'PaperPositionMode','auto');
grid on
set(gca,'linewidth',1.5);
set(gca,'GridLineStyle','-');
set(cvc, 'Position',[0 0 29.7 21]);
set(cvc,'PaperOrientation','landscape');

subplot(1,2,1)
plot(chargeData,currentData,'k');
xlabel('Charge [uAh]');
ylabel('Current [uA]');
title('Current over Charge');
axis tight;
grid on;

subplot(1,2,2)
plot(chargeData,voltData,'k');
xlabel('Charge [uAh]');
ylabel('Voltage [V]');
title('Voltage over Charge');
axis tight;
grid on;

%% Intensity (compensated) over Voltage black & seperate

iov = figure(5);
set(iov,'Units','centimeters');
pos = get(iov,'Position');
set(iov,'PaperPositionMode','auto');
grid on
set(gca,'linewidth',1.5);
set(gca,'GridLineStyle','-');
set(iov, 'Position',[0 0 29.7 21]);
set(iov,'PaperOrientation','landscape');

subplot(1,3,1);
plot(voltData,color_red_cor,'r');
xlabel('Voltage [V]');
ylabel('Optical intensity [arb. units]');
title('Red Channel over Voltage');
axis tight;
grid on;

subplot(1,3,2);
plot(voltData,color_green_cor,'g');
xlabel('Voltage [V]');
ylabel('Optical intensity [arb. units]');
title('Green Channel over Voltage');
axis tight;
grid on;

subplot(1,3,3);
plot(voltData,color_blue_cor,'b');
xlabel('Voltage [V]');
ylabel('Optical intensity [arb. units]');
title('Blue Channel over Voltage');
axis tight;
grid on;

%% Intensity (compensated) over Charge

iocc = figure(6);
set(iocc,'Units','centimeters');
pos = get(iocc,'Position');
set(iocc,'PaperPositionMode','auto');
grid on
set(gca,'linewidth',1.5);
set(gca,'GridLineStyle','-');
```

```
set(iocc, 'Position',[0 0 29.7 21]);
set(iocc,'PaperOrientation','landscape');


subplot(1,3,1);
plot(chargeData,color_red_cor,'r');
xlabel('Charge [uAh]');
ylabel('Optical Intensity [arb. units]');
title('Red Channel over Charge');
axis tight;
grid on;

subplot(1,3,2);
plot(chargeData,color_green_cor,'g');
xlabel('Charge [uAh]');
ylabel('Optical intensity [arb. units]');
title('Green Channel over Charge');
axis tight;
grid on;

subplot(1,3,3);
plot(chargeData,color_blue_cor,'b');
xlabel('Charge [uAh]');
ylabel('Optical intensity [arb. units]');
title('Blue Channel over Charge');
axis tight;
grid on;

%%% Intensity (not compensated) over Charge
%
% iocu = figure(7);
% set(iocu,'Units','centimeters');
% pos = get(iocu,'Position');
% set(iocu,'PaperPositionMode','auto');
% grid on
% set(gca,'linewidth',1.5);
% set(gca,'GridLineStyle','-');
% set(iocu, 'Position',[0 0 29.7 21]);
% set(iocu,'PaperOrientation','landscape');
%
% subplot(1,3,1);
% plot(chargeData,color_red_unc,'k');
% xlabel('Charge [uAh]');
% ylabel('Optical intensity [arb. units]');
% title('Uncorrected Red Channel over Charge');
% axis tight;
% grid on;
%
% subplot(1,3,2);
% plot(chargeData,color_green_unc,'k');
% xlabel('Charge [uAh]');
% ylabel('Optical intensity [arb. units]');
% title('Uncorrected Green Channel over Charge');
% axis tight;
% grid on;
%
% subplot(1,3,3);
% plot(chargeData,color_blue_unc,'k');
% xlabel('Charge [uAh]');
% ylabel('Optical intensity [arb. units]');
% title('Uncorrected Blue Channel over Charge');
% axis tight;
% grid on;

%%% Three dimensional imageTime,intensity,charge
%
% ictd = figure(8);
% set(ictd,'Units','centimeters');
% pos = get(ictd,'Position');
% set(ictd,'PaperPositionMode','auto');
% grid on
% set(gca,'linewidth',1.5);
% set(gca,'GridLineStyle','-');
% set(ictd, 'Position',[0 0 29.7 21]);
% set(ictd,'PaperOrientation','landscape');
%
% x= imageTime + 1;
% y1= color_red_cor;
% y2= color_green_cor;
% y3= color_blue_cor;
% z= chargeData;
%
% plot3(x,y1,z,'r',x,y2,z,'g',x,y3,z,'b');
% xlabel('Time [hrs]');
% ylabel('Optical intensity (corrected)');
% zlabel('Charge [uAh]');
% title('Intensity(Compensated), Charge, Time in 3D');
% axis tight;
% grid on;
%
%%% Three dimensional imageTime,intensity,voltage
%
% ivtd = figure(9);
% set(ivtd,'Units','centimeters');
% pos = get(ivtd,'Position');
% set(ivtd,'PaperPositionMode','auto');
% grid on
% set(gca,'linewidth',1.5);
% set(gca,'GridLineStyle','-');
% set(ivtd, 'Position',[0 0 29.7 21]);
% set(ivtd,'PaperOrientation','landscape');
%
% x= imageTime + 1;
% y1= color_red_cor;
% y2= color_green_cor;
% y3= color_blue_cor;
% z= voltData;
%
% plot3(x,y1,z,'r',x,y2,z,'g',x,y3,z,'b');
% xlabel('Time [hrs]');
% ylabel('Optical intensity (corrected)');
% zlabel('Voltage [V]');
% title('Intensity(Compensated), Voltage, Time in 3D');
% axis tight;
% grid on;

%% Save plots to file.
% Figure handles are in figures(), file names are appended with
      description{}
% filetypes{} contains the filetypes to export

figures=[cit vcci ird cvc iov iocc];
description={'_charge_intensity_over_time'; '
      _voltage_current_charge_intensity_over_time'; '
      _intensity_data_full'; ...
    '_current_voltage_over_charge'; '_intensity_over_voltage';'
         _intensity_com_over_charge'};
filetypes={'pdf','fig'};

for i = 1:length(figures)
    for j=1:length(filetypes)
        set(figures(i), 'InvertHardCopy', 'on');
        filename = strcat(parser.plotDest,description(i),'.',filetypes(j)
            );
        saveas(figures(i),filename{1});
    end;
end;
```

# B.4. m00_profile_test.m

m00_profile_test.m plots calculates the spatial and temporal gradient within each profile and plot the

intensity profile.

```
%% v1.2 2016-10-18 DMMP, VR, AE
%%Removed Plotting function to intensitaetsverlauf_kompensiert

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

%% read inputdata from configplotter.m file
set(0, 'defaultTextInterpreter', 'none');
parser = config;

%% get intensity data as a mat file
A = load(strcat(parser.ImageRGB_data,'.mat'));
filenames = A.imname(:,1);

B = load(strcat(parser.profile_data,'_C.mat'));
intensity_profile = B.exportMatrix;

intensity_profile_um=[1:length(intensity_profile)]/parser.onemm*1000;

%% get the image time stamps to a text file
[status dates] = system(['sed "1{/^#/d;};s/[\t].*//g" ' pwd '/' parser.
      ImageRGB_data]);
disp(dates);

%% save image timestamps in a temporary text file
%fid = fopen('imageTimeStamps.txt', 'wt');
%fprintf(fid, '%s\n', dates);

%%
%  system(['cat ' parser.voltageFile '|grep -v "#"|./TimestampMatcher -f
      ./imageTimeStamps.txt >' parser.tempVolt]);
%  system(['cat ' parser.currentFile '|grep -v "#"|./TimestampMatcher -f
      ./imageTimeStamps.txt >' parser.tempCurrent]);
%  system(['cat ' parser.chargeFile '|grep -v "#"|./TimestampMatcher -f
      ./imageTimeStamps.txt >' parser.tempCharge]);


voltMatrix    = dlmread(parser.tempVolt);
currentMatrix = dlmread(parser.tempCurrent);
chargeMatrix  = dlmread(parser.tempCharge);

charge_min=min(chargeMatrix(:,2));
```

```
charge_max=max(chargeMatrix(:,2));
charge_total=charge_max-charge_min;

chargeMatrix(:,3)=(chargeMatrix(:,2)-charge_min)/charge_total;
%delete (parser.tempVolt,parser.tempCurrent,parser.tempCharge);

%% plot
% try
    unixTime = A.exportMatrix(:,1);
    imageDate = unixTime/86400 + datenum(1970,1,1); % convert back to
        matlab time stamp format
    imageTime = (unixTime-A.exportMatrix(1,1))/3600; % * Time in hours
    sumIntensity_c    = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
        exportMatrix(:,7));

    profile_min=min(intensity_profile(:));      %Min/Max limits for plots
    profile_max=max(intensity_profile(:));

    % Calculate mean of several pixels in the profile
    for i = 1:size(intensity_profile,1)
        profile_mean_1(i,:)=mean(intensity_profile(i,1:60));
        profile_mean_2(i,:)=mean(intensity_profile(i,61:120));
        profile_mean_3(i,:)=mean(intensity_profile(i,121:180));
        profile_mean_4(i,:)=mean(intensity_profile(i,181:200));
    end

    % Calculate the temporal gradient of the profiles
    intensity_profile(size(A.exportMatrix, 1)+1,:)=intensity_profile(size
        (A.exportMatrix, 1),:);
    for i = 1:size(A.exportMatrix, 1)
        profile_temporal_diff(i,:)=intensity_profile(i+1,:)-
            intensity_profile(i,:);
    end
    profile_temporal_diff_min=min(profile_temporal_diff(:));      %Min/
        Max limits for plots
    profile_temporal_diff_max=max(profile_temporal_diff(:));

    % Calculate the spatial gradient within each profile
    for i = 1:size(intensity_profile,2)-1
        profile_spatial_diff(:,i)=intensity_profile(:,i+1)-
            intensity_profile(:,i);
    end
    profile_spatial_diff_min=min(profile_spatial_diff(:));      %Min/Max
        limits for plots
    profile_spatial_diff_max=max(profile_spatial_diff(:));
%% Figure 1
    pot=figure(1);
    step_size=7;
    step_start=27;
    marker={'+','o','*','.','x','s','d','^','v','>','<'};
    colors={[0.1,0.1,0.1],
            [0.2,0.2,0.2],
            [0.3,0.3,0.3],
            [0.4,0.4,0.4],
            [0.5,0.5,0.5],
            [0.6,0.6,0.6]};
    pbaspect([1 0.6 1]);
    hold on;
    for i=1:6
    plot(intensity_profile_um,intensity_profile((i*step_size+step_start)
        ,:)','Color',colors{i},'Marker',marker{i})
    legends{i}=strcat(num2str(chargeMatrix(i*step_size+step_start,3)*100,
        '%10.0f\n'),'% charged');
    end
    hold off;
    pbaspect([1 0.4 1]);
    xlabel('Distance from electrode border [micrometer]');
    ylabel('intensity profile [arb.unit]');
    legend(legends)


    figures=[pot];
    description={'profile_over_time'};
    filetypes={'pdf','fig'};

    for i = 1:length(figures)
        for j=1:length(filetypes)
            set(figures(i), 'InvertHardCopy', 'on');
            filename = strcat(parser.plotDest,description(i),'.',
                filetypes(j));
            saveas(figures(i),filename{1});
        end;
    end;



%% Figure 2: Temporal evolution
    h = figure(2);
    whitebg('w'); %change figure background color

    %figure settings
    %settings:
    plotlinewidth = 1.5;
    gridlinewidth = 1;

    % font size
    % set(findall(h,'type','text'),'FontSize',50,'fontWeight','bold')

    %Position
    set(h,'Units','centimeters');
    pos = get(h,'Position');
    set(h,'PaperPositionMode','auto');
    %set(0,'DefaultAxisFontSize',16)

    %grid-line-width
    grid on
    set(gca,'linewidth',gridlinewidth);
    set(gca,'GridLineStyle','-');

    set(h, 'Position',[0 0 29.7 21]);
    set(h,'PaperOrientation','landscape');
    a = imread(strcat(parser.imageSource,char(filenames(1))));



    % iterate through all images


    for i = 1:size(A.exportMatrix, 1)    %All images


        subplot(3,1,1);


        plot(intensity_profile(i,:))
        axis tight;
        ylim([profile_min profile_max])
        xlabel('Pixel Position');
        ylabel('intensity profile[arb.unit]');
        title('intensity profile over pixel position');
        grid on;

        subplot(3,1,2);


        plot(profile_temporal_diff(i,:))
        axis tight;
        ylim([profile_temporal_diff_min profile_temporal_diff_max])
        xlabel('Pixel Position');
        ylabel('intensity profile[arb.unit]');
        title('temporal differential of the intensity profile over pixel
            position');
        grid on;

        subplot(3,1,3);


        plot(profile_spatial_diff(i,:))
        axis tight;
        ylim([profile_spatial_diff_min profile_spatial_diff_max])
        xlabel('Pixel Position');
        ylabel('intensity profile[arb.unit]');
        title('spatial differential of the intensity profile over pixel
            position');
        grid on;


        drawnow;                                    %# force refresh
        %pause(0.1)                                 %# slow down animation

        %Filename
        set(gcf, 'InvertHardCopy', 'off'); % print to jpg exact colors of
            figure
        filename = strcat(parser.imageDest,'Profileframes/',int2str(i));
        saveas(gcf,filename,'jpg')
        disp(['Part: ' num2str(i) '/' num2str(size(A.exportMatrix, 1))]);
    end

%% Figure 2: Mean of several pixels within each profile
    h2 = figure(2);
    whitebg('w'); %change figure background color

    %settings:
    plotlinewidth = 1.5;
    gridlinewidth = 1;

    % font size
    % set(findall(h,'type','text'),'FontSize',50,'fontWeight','bold')

    %Position
    set(h2,'Units','centimeters');
    pos = get(h2,'Position');
    set(h2,'PaperPositionMode','auto');
    %set(0,'DefaultAxisFontSize',16)

    %grid-line-width
    grid on
    set(gca,'linewidth',gridlinewidth);
    set(gca,'GridLineStyle','-');

    set(h2, 'Position',[0 0 29.7 21]);
```

```
    set(h2,'PaperOrientation','landscape');
    a = imread(strcat(parser.imageSource,char(filenames(1))));



        subplot(4,1,1);


        plot(imageTime,profile_mean_1, 'b')
        axis tight;
        ylim([profile_min profile_max])
        xlabel('image #');
        ylabel('intensity profile[arb.unit]');
        title('temporal differential of the intensity profile over pixel
              position');
        grid on;

        subplot(4,1,2);


        plot(imageTime,profile_mean_2, 'b')
        axis tight;
        ylim([profile_min profile_max])
        xlabel('image #');
        ylabel('intensity profile[arb.unit]');
        title('picture_a');
        grid on;

        subplot(4,1,3);


        plot(imageTime,profile_mean_3, 'b')
        axis tight;
        ylim([profile_min profile_max])
        xlabel('image #');
        ylabel('intensity profile[arb.unit]');
        title('picture_a');
        grid on;

        subplot(4,1,4);

        plot(imageTime,profile_mean_4, 'b')
        axis tight;
        ylim([profile_min profile_max])
        xlabel('image #');
        ylabel('intensity profile[arb.unit]');
        title('picture_a');
        grid on;
%%
    h3 = figure(3);
    whitebg('w'); %change figure background color

    %settings:
    plotlinewidth = 1.5;
    gridlinewidth = 1;

    % font size
    % set(findall(h,'type','text'),'FontSize',50,'fontWeight','bold')

    %Position
    set(h3,'Units','centimeters');
    pos = get(h3,'Position');
    set(h3,'PaperPositionMode','auto');
    %set(0,'DefaultAxisFontSize',16)

    %grid-line-width
    grid on
    set(gca,'linewidth',gridlinewidth);
    set(gca,'GridLineStyle','-');

    set(h3, 'Position',[0 0 29.7 21]);
    set(h3,'PaperOrientation','landscape');
    a = imread(strcat(parser.imageSource,char(filenames(1))));



        plot(imageTime,profile_mean_1, 'b')
        hold on;
        plot(imageTime,profile_mean_2, 'r')
        hold on;
        plot(imageTime,profile_mean_3, 'g')
        hold on;
        plot(imageTime,profile_mean_4, 'k')
        hold off;
        axis tight;
        ylim([profile_min profile_max])
        xlabel('Time [hrs]');
        ylabel('intensity profile[arb.unit]');
        title('intensity evolution over time');
        legend('First Quarter','Second Quarter','Third Quarter','Fourth
              Quarter')
        grid on;


        %pause(0.1)                          %# slow down animation
%%
    system('mogrify -rotate 90 ../Output/Images/Profileframes/*.jpg');
    system('avconv -i ../Output/Images/Profileframes/%d.jpg -c:v mjpeg
           ../Output/Videos/profile_video.avi');
```

# B.5. m00_edge_test.m

m00_edge_test.m Preprocessing of the image such as Edge detection, Section difference, Noise removal, Binarization, Grayscaling are all done in this Matlab file.

```
% v1.2 2016-11-08 15:00 VR

close all;
clear all;
clc;

%
parser = config();

n=1;
offset=4;
step_size=4;

for n=1:5
    section_image1=rgb2gray(imread(strcat(parser.imageDest,'/Section/
          section_1.jpg')));        % every nth picture from the section
          folder is taken
    section=imread(strcat(parser.imageDest,'/Section/section_',num2str(n*
          step_size+offset),'.jpg'));        % every nth picture from the
           section folder is taken
    imwrite(section,strcat(parser.imageDest,'Edge_test/section_',num2str(
          n),'.jpg'));
    section=rgb2gray(section);                    %   parameter for
          binarization: 0.16
    imwrite(section,strcat(parser.imageDest,'Edge_test/section_grayscale_
          ',num2str(n),'.jpg'));
    if n==1
        %section_image1=section;
        section_diff=section;                     %   parameter for
              binarization: 0.44
        section_diff_from_image1=section;         %   parameter for
              binarization: 0.44
    else
        section_diff=(1.2*section-0.8*section_old)*2.5;%((section-
              section_old)+10)*10;
        imwrite(section_diff,strcat(parser.imageDest,'Edge_test/
              section_diff_',num2str(n),'.jpg'));
        section_diff_from_image1=(1.2*section-0.8*section_image1)*2.5;
        imwrite(section_diff_from_image1,strcat(parser.imageDest,'
              Edge_test/section_diff_from_image1_',num2str(n),'.jpg'));

    end;
    figure(10000)
    imshow(section_image1)

    section_noise_removal=medfilt2(section, [20 20]);
    imwrite(section_noise_removal,strcat(parser.imageDest,'Edge_test/
          section_noise_removal_',num2str(n),'.jpg'));
    section_binary=im2bw(section_noise_removal, 0.81);
    imwrite(section_binary,strcat(parser.imageDest,'Edge_test/
          section_binary_',num2str(n),'.jpg'));
    edge_source=section_binary;
    section_edge=imcomplement(edge(edge_source, 'canny', 0.03));
    imwrite(section_edge,strcat(parser.imageDest,'Edge_test/section_edge_
          ',num2str(n),'.jpg'));
    section_edge_log=edge(edge_source, 'log', 0.0001);
    imwrite(section_edge_log,strcat(parser.imageDest,'Edge_test/
          section_edge_log_',num2str(n),'.jpg'));
    section_edge_zerocross=edge(edge_source, 'zerocross', 0.00015);
    imwrite(section_edge_zerocross,strcat(parser.imageDest,'Edge_test/
          section_edge_zerocross_',num2str(n),'.jpg'));


    brightness=sum(sum(section_binary(:)))/402688;
    disp(['File: ',num2str(n),'. Percentage of black pixels: ',num2str
          (100-brightness*100),'%']);



    section_old=section;

end;

%%
figure(2)
imshow(section)
imshow(section_noise_removal)
imshow(section_binary)
imshow(section_edge)
```

```
imshow(section_edge_zerocross)
imshow(section_edge_log)
sum(sum(section_binary()));

%% Charge and Intensity + Images
workflow = figure(1);

% Charge and Intensity over time for red
n=30;

image={'section_';'section_noise_removal_';'section_binary_';'
    section_edge_'};
%image={'section_';'section_edge_';'section_edge_log_';'
    section_edge_zerocross_'}

for n=1:5
% Charge and intensity over time for green
subplot_tight(4,5,n,.01);
imshow(strcat(parser.imageDest,'/Edge_test/',image{1},num2str(n),'.jpg'))
end

for n=1:5
% Charge and intensity over time for green
subplot_tight(4,5,n+5,.01);
imshow(strcat(parser.imageDest,'/Edge_test/',image{2},num2str(n),'.jpg'))
end

for n=1:5
% Charge and intensity over time for green
subplot_tight(4,5,n+10,.01);
imshow(strcat(parser.imageDest,'/Edge_test/',image{3},num2str(n),'.jpg'))
end

for n=1:5
% Charge and intensity over time for green
subplot_tight(4,5,n+15,.01);
imshow(strcat(parser.imageDest,'/Edge_test/',image{4},num2str(n),'.jpg'))
end

% figures=[workflow];
% description={'_edge_detection_workflow'};
% filetypes={'pdf','fig'};
%
% for i = 1:length(figures)
%     for j=1:length(filetypes)
%         set(figures(i), 'InvertHardCopy', 'on');
%         filename = strcat(parser.plotDest,description(i),'.',filetypes(
    j));
%         saveas(figures(i),filename{1});
%     end;
% end;
```

# B.6. m03_videoCreate_v02.m

m03_videoCreate_v02.m create video out of the electrical and optical input data of plot create Matlab

file together with the differential reflectivity profile over time.

```
%% v1.2 2016-10-18 DMMP, VR, AE
%%Removed Plotting function to intensitaetsverlauf_kompensiert

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

%% read inputdata from configplotter.m file
parser = config;
profiling=parser.profiles;

%% Input/Output files


%% get reflectivity data as a mat file
A = load(strcat(parser.ImageRGB_data,'.mat'));
filenames = A.imname(:,1);

if profiling==1;
B = load(strcat(parser.profile_data,'_C.mat'));
intensity_profile = B.exportMatrix;
profile_min=min(intensity_profile(:));
profile_max=max(intensity_profile(:));
intensity_profile_um=[1:length(intensity_profile)]/parser.onemm*1000;
end

voltMatrix    = dlmread(parser.tempVolt);
currentMatrix = dlmread(parser.tempCurrent);
chargeMatrix  = dlmread(parser.tempCharge);

%% plot
% try
    unixTime = A.exportMatrix(:,1);
    imageDate = unixTime/86400 + datenum(1970,1,1); % convert back to
        matlab time stamp format
    imageTime = (unixTime-A.exportMatrix(1,1))/3600; % Time from start in
         hours. If using imageDate, uncomment datetick commands and
        adjust label!
    sumIntensity_c    = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
        exportMatrix(:,7));

    chargeData = chargeMatrix(:,2)/3600*1000*1000;      % Charge in uAh
    voltData = voltMatrix(:,2);                         % Voltage in V
    currentData = currentMatrix(:,2)*1000*1000;         % Current in uA
    color_red_unc = A.exportMatrix(:,2);
    color_green_unc = A.exportMatrix(:,3);
    color_blue_unc = A.exportMatrix(:,4);
    color_red_cor = A.exportMatrix(:,5);
    color_green_cor = A.exportMatrix(:,6);
    color_blue_cor = A.exportMatrix(:,7);
    color_all_cor    = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
        exportMatrix(:,7));



%%Figure Setting
    h = figure(1);
    whitebg('w'); %change figure background color
    plotlinewidth = 1.5;
    gridlinewidth = 1;

    % font size
    % set(findall(h,'type','text'),'FontSize',50,'fontWeight','bold')

    %Position
    set(h,'Units','centimeters');
    pos = get(h,'Position');
    set(h,'PaperPositionMode','auto');
    %set(0,'DefaultAxisFontSize',16)

    %grid-line-width
    grid on
    set(gca,'linewidth',gridlinewidth);
    set(gca,'GridLineStyle','-');

    set(h, 'Position',[0 0 29.7 21]);
    set(h,'PaperOrientation','landscape');
    a = imread(strcat(parser.imageSource,char(filenames(1))));


%%Figure Setting

   %% plot volt graph
    subplot(3,3,[1,2,4,5]);

    imshow(a);

    subplot(4,4,4);
    y1 = voltData;
    plot(imageTime, y1, 'k','LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('U [V]','FontSize',15);
    title('Voltage over time','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coords text
    hLine1 = line('XData',imageTime(1), 'YData',y1(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);
    % hTxt1 = text(imageTime(1), y1(1), sprintf('(%.2f,%.2f)',imageTime
        (1),y1(1)), ...
    % 'Color','b', 'FontSize',8, ...
    % 'HorizontalAlignment','left', 'VerticalAlignment','top');

  %% plot current graph

  %% subplot(2,2,12);
    subplot(4,4,8);

    y2 = currentData;
    plot(imageTime, y2, 'k','LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('I [uA]','FontSize',15);
    title('Current over time','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coordinates
    hLine2 = line('XData',imageTime(1), 'YData',y2(1), 'Color','r', ...
    'Marker','o', 'MarkerSize',5, 'LineWidth',2);


    %% plot charge graph
    %%subplot(3,3,7);
    subplot(4,4,12);
    y3 = chargeData;
    plot(imageTime,y3,'k','LineWidth',2);
```

```
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('Q [uAh]','FontSize',15);
    title('Charge over time','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coordinates
    hLine3 = line('XData',imageTime(1), 'YData',y3(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);

   %% plot reflectivity graph

    subplot(4,4,13);
    y4 = color_red_cor; y5 = color_green_cor; y6 = color_blue_cor;
    plot(imageTime, y4, 'r', imageTime, y5, 'g', imageTime, y6, 'b','
        LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('Reflectivity','FontSize',15);
    title('Reflectivity graph','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coordinates
    hLine4 = line('XData',imageTime(1), 'YData',y4(1), 'Color','y', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);
    hLine5 = line('XData',imageTime(1), 'YData',y5(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);
    hLine6 = line('XData',imageTime(1), 'YData',y6(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);


%% Plot Gradient graph/Differential Reflectivity profile over time
if profiling ~=0;
    subplot(4,4,14);
    plot(chargeData,color_blue_cor)
    axis tight;
    xlabel('Charge [uAh]');
    ylabel('Optical reflectivity');
    title('Reflectivity (blue) over charge');
    grid on;
    hLine8 = line('XData',chargeData(1), 'YData',color_blue_cor(1), '
        Color','r','Marker','o', 'MarkerSize',5, 'LineWidth',2);


else
    subplot(4,4,14);
    plot(chargeData,color_blue_cor)
    axis tight;
    xlabel('Charge [uA]');
    ylabel('Optical reflectivity');
    title('Reflectivity (blue) over charge');
    grid on;
    hLine8 = line('XData',chargeData(1), 'YData',color_blue_cor(1), '
        Color','r','Marker','o', 'MarkerSize',5, 'LineWidth',2);

    subplot(4,4,15);
    plot(voltData,color_blue_cor)
    axis tight;
    xlabel('Charge [uA]');
    ylabel('Optical reflectivity');
    title('Reflectivity over voltage (blue)');
    grid on;
    hLine9 = line('XData',voltData(1), 'YData',color_blue_cor(1), 'Color
        ','r','Marker','o', 'MarkerSize',5, 'LineWidth',2);
end



    %% iterate through all images
    for i = 1:size(A.exportMatrix, 1)    %All images

        subplot(4,4,[1,2,3,5,6,7,9,10,11]);
        filepath = strcat(parser.imageSource,char(filenames(i)));
        a = imread(filepath);
        imshow(a);
        rectangle('Position',[parser.xposl parser.yposu (parser.xposr-
            parser.xposl) (parser.yposd-parser.yposu)], 'LineWidth',3,
             'EdgeColor','r');
        rectangle('Position',[parser.ref_xposl parser.ref_yposu (parser.
            ref_xposr-parser.ref_xposl) (parser.ref_yposd-parser.
            ref_yposu)], 'LineWidth',4, 'EdgeColor','g');
    if profiling == 1;
      rectangle('Position',[parser.profileA_xposl parser.profileA_yposu (
          parser.profileA_xposr-parser.profileA_xposl) (parser.
          profileA_yposd-parser.profileA_yposu)], 'LineWidth',3, '
          EdgeColor','m');
    end
        rectangle('Position',[50 1100 parser.onemm 50], 'LineWidth',3, '
            EdgeColor','w');
        text(parser.ref_xposl+4,parser.ref_yposu+14,'REF','Color','g','
            FontSize',24)
        text(80,1117,'1 mm','Color','w')

        if profiling ~= 0;
            subplot(4,4,[15,16]);
            plot(intensity_profile(i,:))
            %plot(intensity_profile_um,intensity_profile(i,:))
            axis tight;
            ylim([profile_min profile_max])
            xlabel('Distance from electrode border [pixel]');
            ylabel('Reflectivity');
            title('Reflectivity profile');
            grid on;
    else




        subplot(4,4,16);
        b(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr
            ,3);
        hist(double(b(:)),255,'b');
        hold on;
        g(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr
            ,2);

        hist(double(g(:)),255,'g');
        hold on;
        r(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr
            ,1);
        hist(double(r(:)),255,'r');
        hold off;
        title('Histogram','FontSize',20);
        ylabel ('Pixel occurrence','FontSize',20);
        xlabel ('Pixel RGB-value','FontSize',20);
        if i==1
           hist_ylim=1.5*max(hist(double(g(:)),255));       % 50%
               contingency ceiling
        end
        axis([0 255 0 hist_ylim]);
        %axis tight;
        grid on

        h = findobj(gca,'Type','patch');
        set(h(1),'facecolor', 'r','edgecolor','r');
        set(h(2),'facecolor', 'g','edgecolor','g');
        set(h(3),'facecolor', 'b','edgecolor','b');

        set(hLine8, 'XData',chargeData(i), 'YData',color_blue_cor(i))
        set(hLine9, 'XData',voltData(i), 'YData',color_blue_cor(i))
    end

    % Moving bullet along the graph curves
    h = findobj(gca,'Type','patch');

     set(hLine1, 'XData',imageTime(i), 'YData',y1(i))
     set(hLine2, 'XData',imageTime(i), 'YData',y2(i))
     set(hLine3, 'XData',imageTime(i), 'YData',y3(i))
     set(hLine4, 'XData',imageTime(i), 'YData',y4(i))
     set(hLine5, 'XData',imageTime(i), 'YData',y5(i))
     set(hLine6, 'XData',imageTime(i), 'YData',y6(i))
     set(hLine8, 'XData',chargeData(i), 'YData',color_blue_cor(i))


    drawnow;                              %# force refresh
    %pause(0.1)                           %# slow down animation

    set(gcf, 'InvertHardCopy', 'off'); % print to jpg exact colors of
        figure
    filename = strcat(parser.videoframes,int2str(i));
    saveas(gcf,filename,'jpg')
    disp(['Part: ' num2str(i) '/' num2str(size(A.exportMatrix, 1))]);
end

%system('mogrify -rotate 90 ../Output/Images/Videoframes/*.jpg');
%system('avconv -i ../Output/Images/Videoframes/%d.jpg -c:v mjpeg ../
    Output/Videos/video.avi');
```

# B.7. estimate_flow_demo.m

estimate_flow_demo.m does the flow field calculations

```
%% Plot all different sigmas
%
% sigmas=[0.01, 0.5, 1];
clear
close all
%%
%%lambdas=[1, 10, 100];
%lambdas=[1, 5, 10];
lambdas=[3];

methods=['classic+nl-fast', 'hs'];
path='./data/other-data/RubberWhale/MR17Z2P2_prepro/';
```

```
files = dir(fullfile(path, '*.png'));
noFiles  = size(files, 1);
total = length(lambdas)*(noFiles-1)*2;

    for i =1:noFiles-1
        system(['cp -p ' strcat(path,files(i).name) ' ./data/other-data/
            RubberWhale/frame10.png']);
        system(['cp -p ' strcat(path,files(i+1).name) ' ./data/other-data
            /RubberWhale/frame11.png']);
        progress=strcat('Method: Classic+NL, Files:', files(i).name,', ',
             files(i+1).name,', ', num2str((j-1)*noFiles+i), ' out of
            ', num2str(total), ' total done.');
        disp(progress)
        fid = fopen('Output/Plots/notes.txt','at');
        fprintf(fid, '%s\n', progress);
        fclose(fid);
        try
        [path2,filename,ext]=fileparts(files(i+1).name);
        uv = estimate_flow_demo('classic+nl-fast', 4, 'middle-other', '
            lambda', 3,'filename',filename)
        end
    end

%
%
% for j=1:3
%     for i =1:noFiles-1
%         system(['cp -p ' strcat(path,files(i).name) ' ./data/other-data
      /RubberWhale/frame10.png']);
%         system(['cp -p ' strcat(path,files(i+1).name) ' ./data/other-
      data/RubberWhale/frame11.png']);
%         progress=strcat('Method: HS, Lambda = ',num2str(lambdas(j)),'
      Files:', files(i).name,', ', files(i+1).name,', ', num2str(total
      /2+(j-1)*noFiles+i), ' out of ', num2str(total), ' total done.');
%         disp(progress)
%         fid = fopen('Output/Plots/notes.txt','at');
%         fprintf(fid, '%s\n', progress);
%         fclose(fid);
%         try
%         uv = estimate_flow_demo('hs', 4, 'middle-other', 'lambda',
      lambdas(j))
%         end
%     end
% end


disp(['All done. Maybe there was also a horrible error, so all might not
      be done!']);

%
% for i = 1:3
%     for j = 1:3
%         sigma_s_value=sigmas(i);
%         sigma_d_value=sigmas(j);
%         try
%             uv = estimate_flow_demo('classic+nl-fast', 4, 'middle-other
      ', 'lambda',1,'sigma_s', sigmas(i), 'sigma_d', sigmas(j));
%         end
%     end
% end


%% v1.2 2016-10-18 DMMP, VR, AE
%%Removed Plotting function to intensitaetsverlauf_kompensiert

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

%% read inputdata from configplotter.m file
parser = config;
profiling=parser.profiles;

%% Input/Output files


%% get reflectivity data as a mat file
A = load(strcat(parser.ImageRGB_data,'.mat'));
filenames = A.imname(:,1);

if profiling==1;
B = load(strcat(parser.profile_data,'_C.mat'));
intensity_profile = B.exportMatrix;
profile_min=min(intensity_profile(:));
profile_max=max(intensity_profile(:));
intensity_profile_um=[1:length(intensity_profile)]/parser.onemm*1000;
end

voltMatrix     = dlmread(parser.tempVolt);
currentMatrix = dlmread(parser.tempCurrent);
chargeMatrix  = dlmread(parser.tempCharge);

%% plot
% try
    unixTime = A.exportMatrix(:,1);
    imageDate = unixTime/86400 + datenum(1970,1,1); % convert back to
        matlab time stamp format
    imageTime = (unixTime-A.exportMatrix(1,1))/3600; % Time from start in
         hours. If using imageDate, uncomment datetick commands and
        adjust label!
    sumIntensity_c    = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
        exportMatrix(:,7));

    chargeData = chargeMatrix(:,2)/3600*1000*1000;      % Charge in uAh
    voltData = voltMatrix(:,2);                         % Voltage in V
    currentData = currentMatrix(:,2)*1000*1000;         % Current in uA
    color_red_unc = A.exportMatrix(:,2);
    color_green_unc = A.exportMatrix(:,3);
    color_blue_unc = A.exportMatrix(:,4);
    color_red_cor = A.exportMatrix(:,5);
    color_green_cor = A.exportMatrix(:,6);
    color_blue_cor = A.exportMatrix(:,7);
    color_all_cor    = (A.exportMatrix(:,5) + A.exportMatrix(:,6) + A.
        exportMatrix(:,7));



%%Figure Setting
    h = figure(1);
    whitebg('w'); %change figure background color
    plotlinewidth = 1.5;
    gridlinewidth = 1;

    % font size
    % set(findall(h,'type','text'),'FontSize',50,'fontWeight','bold')

    %Position
    set(h,'Units','centimeters');
    pos = get(h,'Position');
    set(h,'PaperPositionMode','auto');
    %set(0,'DefaultAxisFontSize',16)

    %grid-line-width
    grid on
    set(gca,'linewidth',gridlinewidth);
    set(gca,'GridLineStyle','-');

    set(h, 'Position',[0 0 29.7 21]);
    set(h,'PaperOrientation','landscape');
    a = imread(strcat(parser.imageSource,char(filenames(1))));


%%Figure Setting

   %% plot volt graph
    subplot(3,3,[1,2,4,5]);

    imshow(a);

    subplot(4,4,4);
    y1 = voltData;
    plot(imageTime, y1, 'k','LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('U [V]','FontSize',15);
    title('Voltage over time','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coords text
    hLine1 = line('XData',imageTime(1), 'YData',y1(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);
    % hTxt1 = text(imageTime(1), y1(1), sprintf('(%.2f,%.2f)',imageTime
        (1),y1(1)), ...
    % 'Color','b', 'FontSize',8, ...
    % 'HorizontalAlignment','left', 'VerticalAlignment','top');

  %% plot current graph

  %% subplot(2,2,12);
    subplot(4,4,8);

    y2 = currentData;
    plot(imageTime, y2, 'k','LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('I [uA]','FontSize',15);
    title('Current over time','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coordinates
    hLine2 = line('XData',imageTime(1), 'YData',y2(1), 'Color','r', ...
    'Marker','o', 'MarkerSize',5, 'LineWidth',2);


    %% plot charge graph
    %%subplot(3,3,7);
    subplot(4,4,12);
    y3 = chargeData;
    plot(imageTime,y3,'k','LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('Q [uAh]','FontSize',15);
```

```
    title('Charge over time','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coordinates
    hLine3 = line('XData',imageTime(1), 'YData',y3(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);

  %% plot reflectivity graph

    subplot(4,4,13);
    y4 = color_red_cor; y5 = color_green_cor; y6 = color_blue_cor;
    plot(imageTime, y4, 'r', imageTime, y5, 'g', imageTime, y6, 'b','
        LineWidth',2);
    %datetick('x', 'keeplimits');
    xlabel('Time [hrs]','FontSize',15);
    ylabel('Reflectivity','FontSize',15);
    title('Reflectivity graph','FontSize',15);
    axis tight;
    grid on;

    % create moving point + coordinates
    hLine4 = line('XData',imageTime(1), 'YData',y4(1), 'Color','y', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);
    hLine5 = line('XData',imageTime(1), 'YData',y5(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);
    hLine6 = line('XData',imageTime(1), 'YData',y6(1), 'Color','r', ...
        'Marker','o', 'MarkerSize',5, 'LineWidth',2);


%% Plot Gradient graph/Differential Reflectivity profile over time
if profiling ~=0;
    subplot(4,4,14);
    plot(chargeData,color_blue_cor)
    axis tight;
    xlabel('Charge [uAh]');
    ylabel('Optical reflectivity');
    title('Reflectivity (blue) over charge');
    grid on;
    hLine8 = line('XData',chargeData(1), 'YData',color_blue_cor(1), '
        Color','r','Marker','o', 'MarkerSize',5, 'LineWidth',2);


else
    subplot(4,4,14);
    plot(chargeData,color_blue_cor)
    axis tight;
    xlabel('Charge [uA]');
    ylabel('Optical reflectivity');
    title('Reflectivity (blue) over charge');
    grid on;
    hLine8 = line('XData',chargeData(1), 'YData',color_blue_cor(1), '
        Color','r','Marker','o', 'MarkerSize',5, 'LineWidth',2);

    subplot(4,4,15);
    plot(voltData,color_blue_cor)
    axis tight;
    xlabel('Charge [uA]');
    ylabel('Optical reflectivity');
    title('Reflectivity over voltage (blue)');
    grid on;
    hLine9 = line('XData',voltData(1), 'YData',color_blue_cor(1), 'Color
        ','r','Marker','o', 'MarkerSize',5, 'LineWidth',2);
end



    %% iterate through all images
    for i = 1:size(A.exportMatrix, 1)    %All images

        subplot(4,4,[1,2,3,5,6,7,9,10,11]);
        filepath = strcat(parser.imageSource,char(filenames(i)));
        a = imread(filepath);
        imshow(a);
        rectangle('Position',[parser.xposl parser.yposu (parser.xposr-
            parser.xposl) (parser.yposd-parser.yposu)], 'LineWidth',3,
             'EdgeColor','r');
        rectangle('Position',[parser.ref_xposl parser.ref_yposu (parser.
            ref_xposr-parser.ref_xposl) (parser.ref_yposd-parser.
            ref_yposu)], 'LineWidth',4, 'EdgeColor','g');
    if profiling == 1;
      rectangle('Position',[parser.profileA_xposl parser.profileA_yposu (
          parser.profileA_xposr-parser.profileA_xposl) (parser.
          profileA_yposd-parser.profileA_yposu)], 'LineWidth',3, '
          EdgeColor','m');
    end
        rectangle('Position',[50 1100 parser.onemm 50], 'LineWidth',3, '
            EdgeColor','w');
        text(parser.ref_xposl+4,parser.ref_yposu+14,'REF','Color','g','
            FontSize',24)
        text(80,1117,'1 mm','Color','w')

        if profiling ~= 0;
            subplot(4,4,[15,16]);
            plot(intensity_profile(i,:))
            %plot(intensity_profile_um,intensity_profile(i,:))
            axis tight;
            ylim([profile_min profile_max])
            xlabel('Distance from electrode border [pixel]');
            ylabel('Reflectivity');
            title('Reflectivity profile');
            grid on;
    else




        subplot(4,4,16);
        b(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr
            ,3);
        hist(double(b(:)),255,'b');
        hold on;
        g(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr
            ,2);

        hist(double(g(:)),255,'g');
        hold on;
        r(:,:)=a(parser.yposu:parser.yposd,parser.xposl:parser.xposr
            ,1);
        hist(double(r(:)),255,'r');
        hold off;
        title('Histogram','FontSize',20);
        ylabel ('Pixel occurrence','FontSize',20);
        xlabel ('Pixel RGB-value','FontSize',20);
        if i==1
           hist_ylim=1.5*max(hist(double(g(:)),255));        % 50%
               contingency ceiling
        end
        axis([0 255 0 hist_ylim]);
        %axis tight;
        grid on

        h = findobj(gca,'Type','patch');
        set(h(1),'facecolor', 'r','edgecolor','r');
        set(h(2),'facecolor', 'g','edgecolor','g');
        set(h(3),'facecolor', 'b','edgecolor','b');

        set(hLine8, 'XData',chargeData(i), 'YData',color_blue_cor(i))
        set(hLine9, 'XData',voltData(i), 'YData',color_blue_cor(i))
    end

    % Moving bullet along the graph curves
    h = findobj(gca,'Type','patch');

     set(hLine1, 'XData',imageTime(i), 'YData',y1(i))
     set(hLine2, 'XData',imageTime(i), 'YData',y2(i))
     set(hLine3, 'XData',imageTime(i), 'YData',y3(i))
     set(hLine4, 'XData',imageTime(i), 'YData',y4(i))
     set(hLine5, 'XData',imageTime(i), 'YData',y5(i))
     set(hLine6, 'XData',imageTime(i), 'YData',y6(i))
     set(hLine8, 'XData',chargeData(i), 'YData',color_blue_cor(i))


    drawnow;                                %# force refresh
    %pause(0.1)                             %# slow down animation

    set(gcf, 'InvertHardCopy', 'off'); % print to jpg exact colors of
        figure
    filename = strcat(parser.videoframes,int2str(i));
    saveas(gcf,filename,'jpg')
    disp(['Part: ' num2str(i) '/' num2str(size(A.exportMatrix, 1))]);
end

%system('mogrify -rotate 90 ../Output/Images/Videoframes/*.jpg');
%system('avconv -i ../Output/Images/Videoframes/%d.jpg -c:v mjpeg ../
    Output/Videos/video.avi');
```

# B.8. optical_flow_multiple_files.m

run_optical_flow_multiple_files.m plots the Middlebury and vector plots of lambdas parameters.

```
%% Plot all different sigmas
%
% sigmas=[0.01, 0.5, 1];
clear
close all
%%
%%lambdas=[1, 10, 100];
%lambdas=[1, 5, 10];
lambdas=[3];

methods=['classic+nl-fast', 'hs'];
path='./data/other-data/RubberWhale/MR17Z2P2_prepro/';
files = dir(fullfile(path, '*.png'));
noFiles   = size(files, 1);
total = length(lambdas)*(noFiles-1)*2;
```

```
    for i =1:noFiles-1
        system(['cp -p ' strcat(path,files(i).name) ' ./data/other-data/
            RubberWhale/frame10.png']);
        system(['cp -p ' strcat(path,files(i+1).name) ' ./data/other-data
            /RubberWhale/frame11.png']);
        progress=strcat('Method: Classic+NL, Files:', files(i).name,', ',
             files(i+1).name,', ', num2str((j-1)*noFiles+i), ' out of
            ', num2str(total), ' total done.');
        disp(progress)
        fid = fopen('Output/Plots/notes.txt','at');
        fprintf(fid, '%s\n', progress);
        fclose(fid);
        try
        [path2,filename,ext]=fileparts(files(i+1).name);
        uv = estimate_flow_demo('classic+nl-fast', 4, 'middle-other', '
            lambda', 3,'filename',filename)
        end
    end

%
%
% for j=1:3
%     for i =1:noFiles-1
%         system(['cp -p ' strcat(path,files(i).name) ' ./data/other-data
      /RubberWhale/frame10.png']);
%         system(['cp -p ' strcat(path,files(i+1).name) ' ./data/other-
      data/RubberWhale/frame11.png']);
%         progress=strcat('Method: HS, Lambda = ',num2str(lambdas(j)),'
      Files:', files(i).name,', ', files(i+1).name,', ', num2str(total
      /2+(j-1)*noFiles+i), ' out of ', num2str(total), ' total done.');
%         disp(progress)
%         fid = fopen('Output/Plots/notes.txt','at');
%         fprintf(fid, '%s\n', progress);
%         fclose(fid);
%         try
%         uv = estimate_flow_demo('hs', 4, 'middle-other', 'lambda',
      lambdas(j))
%         end
%     end
% end


disp(['All done. Maybe there was also a horrible error, so all might not
      be done!']);

%
% for i = 1:3
%     for j = 1:3
%         sigma_s_value=sigmas(i);
%         sigma_d_value=sigmas(j);
%         try
%             uv = estimate_flow_demo('classic+nl-fast', 4, 'middle-other
      ', 'lambda',1,'sigma_s', sigmas(i), 'sigma_d', sigmas(j));
%         end
%     end
% end
```

# 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, April 20, 2017

City, Date

sign