



TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies ■

Software development of a computer aided method to capture and determine crack lengths of fracture mechanics test specimens

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics

Study branch: 3906T001 – Mechatronics

Author: **Bc. Michal Dostálek**

Supervisor: Prof. Dr. rer. nat. Stefan Bishoff





Assignment for the master thesis

Course of studies: Mechatronics

Student's name: Michal Dostalek

Subject:

Software development of a computer aided method to capture and determine crack lengths of fracture mechanics test specimens

Assignment:

- Selection of a suitable image acquisition software
- Implementation of existing codes
- Development and implementation of codes to determine the crack lengths
- Fusion of the single codes into one programme
- Validation of the software on fracture mechanics test specimens

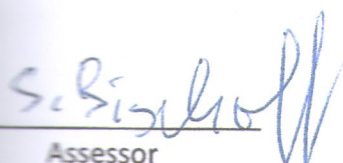
Assessor: Prof. Dr. rer. nat. Stefan Bischoff (Hochschule Zittau/Görlitz)

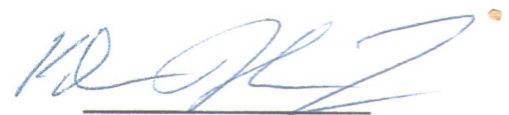
Supervisor: M.A. Dipl.-Ing. (FH) Stefan Keck (Hochschule Zittau/Görlitz)

Date of issue: 01.04.2016

Date of submission: 31.07.2016

Registry-Number.: MA/EMIm14 – 06/ 16


Assessor


Dean

Declaration

I hereby certify that I have been informed that Act 121/2000, the Copyright Act of the Czech Republic, namely Section 60, Schoolwork, applies to my master thesis in full scope. I acknowledge that the Technical University of Liberec (TUL) does not infringe my copyrights by using my master thesis for TUL's internal purposes.

I am aware of my obligation to inform TUL on having used or licensed to use my master thesis in which event TUL may require compensation of costs incurred in creating the work at up to their actual amount.

I have written my master thesis myself using literature listed therein and consulting it with my supervisor and my tutor.

I hereby also declare that the hard copy of my master thesis is identical with its electronic form as saved at the IS STAG portal.

Date: 13.1.2017

Signature: 

Acknowledgements

I would like to express my sincere gratitude M.A. Dipl.-Ing.(FH) Stefan Keck for the continuous support and innovative ideas.

I would like to thank my supervisor Prof. Dr. rer. nat. Stefan Bishoff for his insightful guidance.

Abstract

Cracks are important indicators reflecting the material properties and their effects on the resulting constructions. This thesis presents an automatic crack capturing, detection and classification methodology for a crack test stand. The algorithm is written in C/C++. The complete graphical user interface was created as well.

The test samples are captured and stored with use of complementary metal-oxide-semiconductor industrial cameras. The captured samples are converted into a grey-scale images and areas with possible cracks are found by utilizing morphological image processing techniques and thresholding operations.

Cracks are classified and measured with the help of a scaling factor and mathematical elementary functions. The classification methods were tested on real samples with success over 95%. The experimental results revealed the influence of parameter settings and filter selection and also proved that the proposed approach is effective for automatic crack detection and classification.

The testing algorithm is universal and usable for all sorts of materials from plastics, metals to wood and composites. The program was developed and tested using also fibre materials with epoxide core. Other samples were e.g materials which were tested for their suitability for use in industry.

Part of this thesis also covers the basic principles used in image processing.

Keywords: Computer vision, crack classification, fracture mechanics, crack detection

Contents

| | | |
|----------|---------------------------------------|-----------|
| 1 | Introduction | 13 |
| 2 | Digital image processing | 15 |
| 2.1 | Image formats | 15 |
| 2.1.1 | Lossy compression | 16 |
| 2.1.2 | Lossless compression | 19 |
| 2.2 | Image processing techniques | 20 |
| 2.2.1 | Thresholding | 20 |
| 2.2.2 | Image smoothing | 22 |
| 2.3 | Template matching | 25 |
| 2.4 | Morphological operators | 26 |
| 2.4.1 | Dilation | 26 |
| 2.4.2 | Erosion | 27 |
| 2.4.3 | Opening | 28 |
| 2.4.4 | Closing | 29 |
| 3 | Image acquisition | 30 |
| 3.1 | Hardware | 31 |
| 3.1.1 | Camera | 31 |
| 3.1.2 | Camera lens | 32 |
| 3.1.3 | Filter | 33 |
| 3.2 | Software | 37 |
| 3.2.1 | WebCam | 37 |
| 3.2.2 | uEye | 39 |
| 3.2.3 | Final program | 44 |
| 4 | Image processing | 47 |
| 4.1 | OpenCV 3.0.0 | 47 |
| 4.2 | Pre-processing | 48 |

| | | |
|-------------------|---|-----------|
| 4.2.1 | Reading of a scaling factor | 48 |
| 4.2.2 | Thresholding method algorithm | 51 |
| 4.2.3 | Morphological method algorithm | 53 |
| 4.3 | Processing | 59 |
| 4.4 | Post-processing | 64 |
| 5 | Result presentation | 67 |
| 6 | Conclusion | 70 |
| Appendix A | Contents of enclosed DVD | 71 |
| Appendix B | Changelog of Capture program | 72 |
| Appendix C | Changelog of CrackAnalyzer program | 73 |

List of Figures

| | | |
|-------------|--|----|
| Figure 1.1 | Proposed approach flowchart | 14 |
| Figure 2.1 | Resulting image | 19 |
| Figure 2.2 | Dilation | 27 |
| Figure 2.3 | Erosion | 28 |
| Figure 2.4 | Opening | 29 |
| Figure 2.5 | Closing | 29 |
| Figure 3.1 | CIE 1931 diagram | 34 |
| Figure 3.2 | Properties of the used filters | 34 |
| Figure 3.3 | Without a filter | 35 |
| Figure 3.4 | Blue filter with high light intensity - Zenith blue | 35 |
| Figure 3.5 | Orange filter with high light intensity | 35 |
| Figure 3.6 | Orange filter with low light intensity | 35 |
| Figure 3.7 | Combination of blue and orange filter with low light intensity | 35 |
| Figure 3.8 | Blue filter with high light intensity - Regal blue | 36 |
| Figure 3.9 | Orange filter with high light intensity - Cool LED Orange | 36 |
| Figure 3.10 | Orange filter with low light intensity - Cool LED Orange | 36 |
| Figure 3.11 | Green filter with low light intensity | 36 |
| Figure 3.12 | WebCAM flowchart of used algorithm | 42 |
| Figure 3.13 | uEye flowchart of used algorithm | 43 |
| Figure 3.14 | Capturing program | 44 |
| Figure 3.15 | Capturing program settings | 45 |
| Figure 3.16 | Simplified capture program flow diagram | 46 |
| Figure 4.1 | Image for taking the scaling factor | 49 |
| Figure 4.2 | Thresholded image with template matching | 50 |
| Figure 4.3 | Input data from camera | 51 |
| Figure 4.4 | Threshold Otsu | 52 |
| Figure 4.5 | Threshold Gauss | 52 |
| Figure 4.6 | Cross (left) and Diamond (right) kernel | 53 |

| | | |
|-------------|---|----|
| Figure 4.7 | Input data from camera | 53 |
| Figure 4.8 | Gradient of the image | 54 |
| Figure 4.9 | Erosion | 55 |
| Figure 4.10 | Dilatation | 56 |
| Figure 4.11 | Contrast stretching and saturation | 57 |
| Figure 4.12 | Preprocessed image after thresholding operation | 58 |
| Figure 4.13 | Segmentation after template matching | 59 |
| Figure 4.14 | Segmentation filter | 60 |
| Figure 4.15 | Segmented image after the use of modified mean filter | 61 |
| Figure 4.16 | Crack reconstruction | 62 |
| Figure 4.17 | Reconstructed path | 63 |
| Figure 4.18 | Output image | 65 |
| Figure 5.1 | Darker image | 68 |
| Figure 5.2 | Brighter image | 68 |
| Figure 5.3 | Threshold of darker image | 68 |
| Figure 5.4 | Threshold of brighter image | 68 |
| Figure B.1 | Changelog: Capture program | 72 |

List of Tables

| | | |
|-----|--|----|
| 3.1 | Structure containing data about camera | 39 |
| 3.2 | Used structures | 40 |
| 4.1 | Properties | 65 |
| 4.2 | Output file | 66 |

List of abbreviations

| | |
|--------|---|
| ADC | Analog to Digital Converter. 31 |
| AIO | Asynchronous I/O. 31 |
| BSD | Berkeley Software Distribution. 47 |
| CCD | Charge-coupled Device. 30, 32, 70 |
| CCOEFF | Cross correlation coefficient. 25, 26 |
| CGM | Computer Graphics Metafile. 19 |
| CMOS | Complementary Metal oxide semiconductor. 30, 32 |
| CPU | Central Processing Unit. 13 |
| CUDA | Compute Unified Device Architecture. 47, 48 |
| DCT | Discrete Cosine Transform. 16 |
| EEPROM | Electrically Erasable Programmable Read-Only Memory. 39 |
| GIF | Graphics Interchange Format. 18 |
| GPU | Graphics Processing Unit. 13, 48 |
| IDE | Integrated Development Environment. 47 |
| INDE | Integrated Native Developer Experience. 47 |
| IPP | Integrated Performance Primitives. 47 |
| JPEG | Joint Photographic Experts Group. 16, 18, 19 |
| MDL | Minimal Description Length. 24 |
| MMX | Multi Media Extension. 47 |
| NCC | Cross correlation. 25 |
| OS | Operating System. 47 |
| OSAD | Optimized Sum of Absolute differences. 25 |
| PNG | Portable Network Graphics. 18 |
| SAD | Sum of Absolute differences. 25 |
| SDK | Software Development Kit. 39, 70 |

| | |
|------|---|
| SHD | Sum of Hamming Distances. 25 |
| SI | Système international d'unités, International System of Units. 64 |
| SPD | Spectral Power Distribution. 33 |
| SSD | Sum of Squared Differences. 25 |
| SSE | Streaming SIMD Extensions. 47 |
| SVG | Scalable Vector Graphics. 19 |
| SW | Software. 13 |
| TBB | Threading Building Blocks. 47 |
| USB | Universal Serial Bus. 15, 31, 39 |
| VGA | Video Graphics Array. 40 |
| WVGA | Wide Video Graphics Array. 31 |

1 Introduction

1. **Detection rate** is the most important aspect. If the small cracks can be only partially detected or they are not detected at all, the classification and evaluation is pointless and the equipment or the arrangement of the test stand has to be changed.
2. **Detection accuracy:** under the assumption that we have a high detection rate, the crack detection accuracy must be acceptable, in other words objects which could be misidentified must be removed by the use of SW (Software) filters or another non complex algorithms.
3. **Detection efficiency.** As mentioned above there can be over thousands of acquired images. The processing of such huge number of samples is a challenging task for the CPU (Central Processing Unit) or even for a GPU (Universal Serial Bus). The efficiency of image processing must be high enough, which makes complex algorithms not suitable for this application.

The requirements have to be carefully followed. To meet the detection rate condition it is necessary to have as much detailed images as possible. Therefore the images have to be stored with a high resolution and contrast. The best parameters can be achieved only by using a high quality capturing system (cameras, lenses, filters etc.). The description of the used equipment is in section 3.1. However, the equipment can get expensive and it is always necessary to evaluate the relationship between costs and effectiveness. Moreover, this relationship is non-linear and the effectiveness of the detection rate can be partially overtaken by software methods (both increased and decreased but always followed with a loss of computational power).

The main idea is to work with grey-scale images with cracks represented as brighter areas which should be accentuated. This operation requires the known mean value of the whole grey-scale image to allow contrast stretching and brightness adjustments. In the next step, various threshold filters with different parameters are applied, depending on the properties of the original image. But because this

principle is very light dependent and with higher light intensity whiter areas are created, and the grey-scaled image is devalued. In such case it is possible to use the image from the uEye camera in 24 bit depth mode. The light also creates a brighter area but with the color of the used filter together with contrast stretching in this color channel it is possible to decrease the influence of the light on the crack evaluation.

By the use of these methods the crack is accentuated and separated from the background. But in the image are still small irrelevant objects present. They can be filtered out by means of small matrix filters or by the use of segmentation.

In attempt to distinguish the crack from the other irrelevant objects similar to the crack, future extraction of the real crack becomes a key problem due to the fact it must be simple and quick to process. In this thesis is one of possible methods to do such operations (figure 1.1).

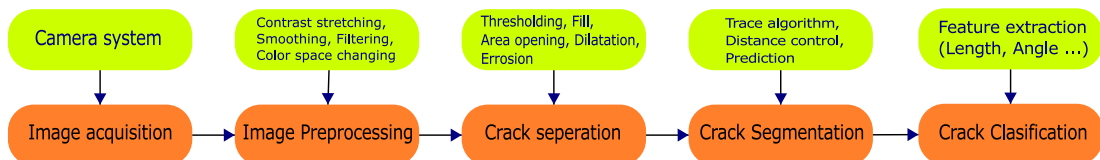


Figure 1.1: Proposed approach flowchart

This thesis is divided into several chapters. The main focus of the first chapter is theory about image processing and image acquisition. Subsequently, the software development for acquiring and processing (including some algorithms) is described. At the end there are presented results with different hardware set-ups such as light intensity and filter type and some recommendation for the application of the developed algorithms.

2 Digital image processing

2.1 Image formats

There exists hundreds of file formats that are used when manipulating with an image. The format of the image should be chosen depending on the application. For example in this thesis the figures are created with vector graphic and the images used for evaluating the cracks are raster images. Those two formats have major differences. Vector images can not be created with a capturing device such as camera, because the data in a vector file contains information about the vectors but the camera sensors capture a matrix. From this definition it is possible to assume why it is necessary to have raster graphic for the camera data. Raster image formats contain large amount of data and therefore they are memory demanding. This becomes a problem when trying to transfer the file via the USB (Universal Serial Bus) to the memory in a short time. This is a place for image compression which can significantly decrease the image size but at the same time it can affect the image quality. The used uEye cameras have the possibility to set compression parameters internally with the option to make additional compression in the PC. The web cams do not allow such setting but data streams are usually very lower so the size is not the concern.

The ideal case would be a bitmap file format with lossless compression, small size of the resulting image and very fast algorithm for handling the file. In this chapter are several file formats presented, together with the compression algorithms and comparison of different methods.

As mentioned before it is possible to divide image formats into two main categories:

- **Lossless compression:** algorithms reduce file size while preserving a perfect copy of the original uncompressed image. Lossless compression generally results in larger files than lossy compression although it is not a rule. Lossless

compression should be used to avoid accumulating stages of re-compression when editing images.

- **Lossy compression:** algorithms preserve a representation of the original uncompressed image that may appear to be a perfect copy but it is not. Lossy compression is able to achieve smaller file sizes compared to lossless compression. Most lossy compression algorithms allow variable compression ration that trades image quality for file size.

2.1.1 Lossy compression

JPEG

JPEG is a method for lossy compression of digital images and it is the most common format when in comes to photographic images. The compression steps of the JPEG are as follows:

1. **Division** of the image into 8x8 blocks of pixels.
2. **Transformation into $Y C_b C_r$ color space.**

$Y C_b C_r$ color space is represented as luminance (Y) and and blue-difference and red-difference chroma components C_b and C_r . The components of $Y C_b C_r$ space are less correlated than the RGB components. Also the human eye is more sensitive to luminance than chrominance. Therefore it is possible to neglect more changes in the chrominance without affecting the quality of the picture. The transformation is depicted in equation 2.1 [2].

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.29900 & 0.58700 & 0.11400 \\ -0.16874 & -0.33126 & 0.50000 \\ 0.50000 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.1)$$

3. **The discrete cosine transform.** The compression baseline process is a mathematical transformation DCT (Discrete Cosine Transform) shown in equation 2.2. The formula is used on square matrix with dimension of N (as mentioned before N mostly equals 8 because larger block takes a longer time and even thought they offer higher quality compression, the trade-off is unreasonable). The output is a square matrix with the same dimension N filled with

frequency coefficients $DCT(i, j)$. [10]

$$DCT(i, j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} pixel(x, y) \cos \left[\frac{(2x+1)i\pi}{2N} \right] \cos \left[\frac{(2y+1)j\pi}{2N} \right]$$

with:

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } w = 0 \\ 1 & \text{if } w > 0 \end{cases} \quad (2.2)$$

4. **Quantization.** Calculated coefficient are real number which has to be saved as integers. Thus it is necessary to change the number type but instead just rounding the coefficients they are first divided by quantization factor to emphasize certain frequencies. The human is eye less sensitive to rapid variations in the image and higher frequencies can be suppressed by choosing higher quantization factor for these frequencies. There exist recommended values for both luminance and chrominance coefficients in the JPEG standard [10].

$$round\left(\frac{DCT(i, j)}{Q(i, j)}\right) \quad (2.3)$$

The JPEG compression process has option to control the quality of the compressed image. Let the parameter q be defined as integer value (given by user) used for calculation of parameter α which is used in the compression process.

$$q = 1, 2, \dots, 98, 100$$

$$\alpha = \begin{cases} \frac{50}{q} & \text{if } 1 \leq q \leq 50 \\ 2 - \frac{50}{q} & \text{if } 50 \leq q \leq 100 \end{cases} \quad (2.4)$$

The defined α parameter is included into equation 2.5 yielding

$$round\left(\frac{DCT(i, j)}{\alpha Q(i, j)}\right) \quad (2.5)$$

The higher the value of q , the lower is α . In the rounding process is information getting lost and the size of the image decreases.

5. **Image reconstruction** The DCT coefficients must be reconstructed, from there is the YC_bC_r vector found with use of IDCT (Inverse Discrete Cosine

Transform) and finally is the RGB vector calculated by inverting the color space transform.

JPEG 2000

The JPEG algorithm leads to possible discontinuities on the borders of the 8x8 blocks which can result in blocking artifacts. Moreover the JPEG only allows one value of resolution when recovering the image. To improve the JPEG algorithm the JPEG 2000 was introduced in 2000. The main difference is in the used mathematical function. In this case a wavelet transform instead of the DCT is used.

The dyadic decomposition

JPEG and JPEG 2000 have similar compression ranges but JPEG 2000 takes more time and computational effort. The blocking artifacts effect is not present in the JPEG 2000 but the images become blurry when the compression ratio is set too high.

GIF

The images in GIF (The Graphics Interchange Format) are compressed using the LZW (Lempel–Ziv–Welch) lossless data compression and the format supports 8-bits per pixel, allowing only 256 colors from 24-bit RGB color space [9]. This makes the format suitable for fast-loading visuals and animation which are also supported by the GIF format.

PNG

PNG (Portable Network Graphics) format supports lossless data compression. PNG was created as a non-patented alternative for GIF. As stated in the format specification [6] the 24-bit RGB and 32-bit RGBA color palettes are supported as well as gray-scale images (with or without alpha channel) and full-color non-palette-based RGB.

2.1.2 Lossless compression

SVG

SVG (Scalable Vector Graphics) is an XML-based vector image format for 2D graphics with animation support. SVG images are defined in XML files (see code 2.1 created with the aid of the format specification [4] and the resulting image on figure 2.1). The files can be searched, indexed and scripted by using a text editor. It is, however, more common to edit SVG images with drawing software.

Code 2.1: XML code for SVG graphic

```
<!DOCTYPE html>
<html>
<body>
<svg height="100" width="100">
  <circle cx="50" cy="50" r="40"
    stroke="black"
    stroke-width="3" fill="red" />
</svg>
</body>
</html>
```

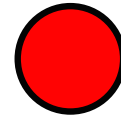


Figure 2.1: Resulting image

Other lossless formats are CGM, Gerber (RS-274x) and PPT

After consideration of all the facts in this chapter is format JPEG selected for capturing the frames for crack analysis. The size of acquired image is in average 100kB. During the image processing are the pictures saved in JPEG 2000 using lossless compression. The resulting format is again JPEG.

2.2 Image processing techniques

2.2.1 Thresholding

When identifying an object or other relevant information of an image, segmentation is usually the first step in this process. Image segmentation divides the original picture into a set of regions corresponding to individual objects, parts or surfaces. One of the simplest method of image segmentation is thresholding. It is effective method how to separate objects especially when the grey levels of object pixels are different from the grey levels of the background.

Thresholding in non-linear operation converting a grey-scaled image into a binary image. The basic application assigns one of two levels to every pixels, depending on its previous intensity. For effective use of this process in image processing application the threshold must be automatically selected. Threshold methods can be split into several categories [15].

- **Histogram shape-based method**

The peaks and values of the histogram determinate the threshold value. For example if there are two dominant peaks in the image histogram the threshold value can lie between those peaks. The pixels are handled independently of their surrounding and the new pixel value is obtained by simple comparing the pixel intensity with threshold value T .

$$g(x, y) = \begin{cases} 0 & \text{if } (x, y) < T \\ 1 & \text{if } (x, y) \geq T \end{cases} \quad (2.6)$$

- **Clustering-based method**

The grey-scale levels are clustered into two parts (background and foreground). The method can be further divided as follows [3]:

- Iterative-based methods

The basic steps for iterative thresholding can be described as follows.

1. Estimation initial threshold value T
2. Dividing of the histogram into two parts $P1$ and $P2$ using T
3. Calculation of mean intensity of the histogram parts $P1$ and $P2$ as μ_1 and μ_2

4. Setting if the new threshold (equation 2.7).

$$T = \frac{\mu_1 + \mu_2}{2} \quad (2.7)$$

5. Repeat steps 2 to 4 until the mean values μ_1 and μ_2 do not change

This method was described by Calvard et al. [1] and it has several modifications which are increasing efficiency.

– Clustering thresholding

The most used and known clustering thresholding method is Otsu's method (detailed described in ...)

– Minimal error thresholding

– Fuzzy clustering thresholding

- **Entropy-based method**

Entropy is a part of the second law of Thermodynamics. It measures spontaneous dispersal of energy. Kapur et al. [12] used the following approach. The background and the foreground are considered as two different signals. Each has its entropy evaluated and summed. In the sum maximum is the threshold value optimal.

- **Object attribute-based method**

This method use similarity measure to extract a threshold value between the original and binarized image. Example of method falling into this category is moment preserving thresholding [19].

- Others, including e.g. the spatial methods which use high-order probability distribution together with correlation or local methods to adapt the threshold on individual pixels.

Otsu's method

Otsu's thresholding method is one of the clustered-based thresholding algorithms. It is assumed that the image has bimodal histogram (containing two dominant peaks) and the pixels can be separated into two classes: foreground and background. The algorithm finds the optimum point of separation of those two classes. The threshold value is calculated by minimizing the weighted within-class variance (equation 2.8):

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad (2.8)$$

where σ_1^2 is variance of the pixels in the background and σ_2^2 is the variance of the pixels in the foreground.

The process of minimizing the weighted within-class variance is equivalent to maximizing between-class variance, in other words, maximizing the distance between the clusters therefore separating the objects from background. The class probabilities q_1 , q_2 and class means μ_1 , μ_2 are defined as follows (equations 2.9):

$$\begin{aligned} q_1(t) &= \sum_{i=1}^I P(i) & \text{and} & & q_2(t) &= \sum_{i=t+1}^t P(i) \\ \mu_1(t) &= \sum_{i=1}^t \frac{iP(i)}{q_1(t)} & \text{and} & & \mu_2(t) &= \sum_{i=t+1}^I \frac{iP(i)}{q_2(t)} \end{aligned} \quad (2.9)$$

The individual class variances are shown in the following equations 2.10.

$$\begin{aligned} \sigma_1^2(t) &= \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \\ \sigma_2^2(t) &= \sum_{i=t+1}^I [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)} \end{aligned} \quad (2.10)$$

This method shows good results when the histogram bimodal condition is satisfied. In case that the object is significantly smaller than the background area the histogram loses its bimodality and the method becomes less effective. Furthermore if there is a additive noise present in the image the valleys of the histogram are degraded and the optimal threshold can not be reached. The program with additive noise can be solved using improved two-dimensional Otsu's method.

2.2.2 Image smoothing

Image noise is an undesirable side effect of an image capture produced by camera sensors. This unwanted signal adds spurious information and usually it is removed before more image processing steps take place (e.g. image smoothing before edge detection significantly improves the result).

Mean filter

Mean filtering is an easy to implement method for smoothing images. Each pixel in the image is replaced with the average value of its neighbours (including itself). This leads to elimination of the pixels which do not fit into their surroundings.

The mean filter is a convolution filter with kernel of usually size 3x3 (equation 2.11). Larger kernels are used for more intensive blurring.

$$\text{Mean filter kernel} = \begin{array}{|c|c|c|} \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \end{array} \quad (2.11)$$

Gaussian filter

Gaussian blur is one of the most used effects suppressing image noise and details. The blurring is done by use of Gaussian function (called normal distribution function in statistics) shown in equation 2.12. For use on two dimensional image the two dimensional Gaussian function has to be calculated as the product of two 1D Gaussian functions (equation 2.13). The x stands for distance from the origin in the horizontal axis, similarly y represents the distance in the vertical axis and σ is the width of the Gaussian kernel.

$$1\text{D} : \quad G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (2.12)$$

$$2\text{D} : \quad G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.13)$$

Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel (An example of 5x5 Gaussian kernel is shown in equation 2.14). Then they are summed up to produce the output array. The center value has the largest value which is decreasing symmetrically as the distance from origin increases. This results in gentle smoothing and edge preservation.

$$\text{Gaussian kernel} = \frac{1}{273} \cdot \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array} \quad (2.14)$$

Gaussian adaptive filter

Effective smoothing of the noise can be achieved by using a Gaussian filter with large variance. If the filter is applied on the part of image with abrupt changes in pixel brightness there can also occur side effects such as edge position displacement or edge vanishing. This is problem solvable by adapting the Gaussian filter to the parameters of pixel in the local area. In other words some parts of the image can be more noisy and thus need more smoothing.

Let $I_{\sigma_1}, I_{\sigma_2}, \dots, I_{\sigma_n}$ be smoothed images calculated as convolution of the original image I_0 with a series of increasing Gaussian kernels $G_\sigma, \sigma = \sigma_1, \dots, \sigma_n$ as

$$I_\sigma = I_0 * G_\sigma \quad (2.15)$$

The goal is to find appropriate σ from the scale-space at each x, y pair. The criterion is based on the MDL (Minimal Description Length). A Gaussian filtering can be seen as:

$$I_0(x, y) = \underbrace{I_\sigma(x, y)}_{\text{Low-pass}} + \underbrace{\varepsilon_\sigma(x, y)}_{\text{Residual}} \quad (2.16)$$

where the ε_σ is the difference between the original image and the smoothed image. Rewriting the equation 2.16 with description length results in:

$$dl_{I_0(x, y)} = dl_{I_\sigma(x, y)} + dl_{\varepsilon_\sigma(x, y)} \quad (2.17)$$

The description length of I_σ can be estimated by using Scaling-Uncertainty Principle. The description length of ε_σ can be obtained with use of Central Limit Theorem. Combining the results yields the final algorithm:

$$\hat{I}(x, y) = I_0(x, y) * e^{-\frac{x^2+y^2}{2\sigma^*(x, y)}} \quad (2.18)$$

2.3 Template matching

Template matching is a technique to find areas in a source data pattern that matches an another data pattern. There are several method for template matching. Every template matching method can be also calculated as a normalized function. Primary concept of template matching is to slide a template across the source image and look for similarities. This will be done by calculating the metric at each point, location and pixel. The result of this operation is an information about how good or bad the match was. In practice we use the function `minmaxLoc()` to find the highest value in a map created by the template matching function. In OpenCV 3.0.0 are several methods implemented for this kind of function, but in for the means of this thesis only two of them are used:

- SAD (Sum of Absolute differences)
- OSAD (Optimized Sum of Absolute differences)
- SSD (Sum of Squared Differences)
- NCC (Cross correlation) Cross correlation is algorithm based on finding a position of biggest match by taking a smaller sample (template) and a larger dataset (image), the template will be overlaid and for each pixel in will be a value calculated by multiplying. Then a sum will be calculated and the template will be slid.

Normalized cross correlation is a well-known method for pattern finding. Template T is correlated in opposition to an image X . The correlation at each point of the image. Then the highest score for overlaying will be chosen as a winner.

$$R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}} \quad (2.19)$$

- SHD (Sum of Hamming Distances)
- CCOEFF (Correlation coefficient) It is the same basic framework like NCC but with a different underlying calculation. This method matches a template relative to its mean against the image relative mean, so a perfect match will be the value 1 and a perfect mismatch will be the value -1. The value 0 means

there is no correlation.

$$R(x, y) = \sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))$$

where

$$\begin{aligned} T'(x', y') &= T(x', y') - \frac{1}{w \cdot h} \cdot \sum_{x'', y''} T(x'', y'') \\ I'(x + x', y + y') &= I(x + x', y + y') - \frac{1}{w \cdot h} \sum_{x'', y''} I(x + x'', y + y'') \end{aligned} \quad (2.20)$$

And for the Normed CCOEFF:

$$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}} \quad (2.21)$$

2.4 Morphological operators

Morphological operators offers local pixel transformation for processing region shapes usually used for binary images. The transformations are based on the comparison of pixel surrounding using a patters (structuring element). The operations can be used for smoothing of region boundaries before shape analysis or for removing noise from the picture (caused e.g. by imperfect segmentation). Main operations are dilation and erosion are discussed in the following sections. Those operations are used for definition of more complex morphological operators such as closing and opening which are also further discussed in this section.

In the following definition is binary image $I(u, v)$ defined as a set of pixel location in the foreground.

$$Q_I = \{(u, v) \mid I(u, v) = 1\} \quad (2.22)$$

For the simplification is variable \mathbf{p} defined to represent the coordinate pair $\mathbf{p} = (u, v)$ and the equation 2.22 becomes equation 2.23.

$$Q_I = \{p \mid I(p) = 1\} \quad (2.23)$$

2.4.1 Dilation

Dilation expands and thickens objects in binary images. The output of this operation is controlled by the shape of the used structure element. Let the object A be the picture which should be dilated and object B the structure element. A dilation of

A by B can be described as following set of operation (equation 2.24 [5]).

$$A \oplus B = \{(\mathbf{p} + \mathbf{q}) \mid p \in A, \mathbf{q} \in H\} \quad (2.24)$$

The structuring element has a defined anchor point which usually lies in the center of the kernel. The kernel is scanned over the image and the output pixel value is the maximum value of all the pixels in the input pixel surrounding. When applying the

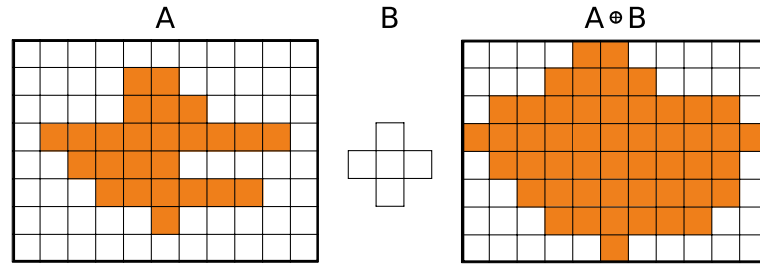


Figure 2.2: Dilation

dilation it is necessary to provide sufficiently large image domains so that operation can not “fall off” the edges. The dilation is commutative (structuring element and the image can switch roles, relation 2.25) as well as associative (big structures can be break into smaller ones, relation 2.26).

$$A \oplus B = B \oplus A \quad (2.25)$$

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C \quad (2.26)$$

2.4.2 Erosion

Erosion operation performs thinning and constriction of the object in binary image (fig 2.3). The output of the operation is controlled by the structuring element. Let the object A be the picture which should be eroted and object B the structure element. A erosion of A by B can be described as following set of operation (equation 2.27 [5]).

$$A \ominus B = \{(\mathbf{p} \in \mathbb{Z}) \mid (\mathbf{p} + \mathbf{q}) \in A, \text{ for every } \mathbf{q} \in H\} \quad (2.27)$$

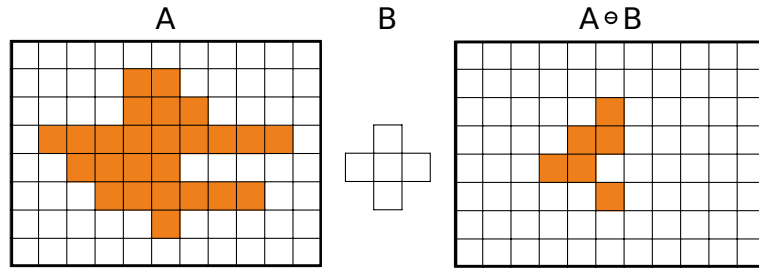


Figure 2.3: Erosion

In opposite to dilation operation erosion is neither commutative (eq. 2.28) nor associative however the following relation holds (eq. 2.29).

$$A \ominus B \neq B \ominus A \quad (2.28)$$

$$(A \ominus B) \ominus C = A \ominus (B \oplus C) \quad (2.29)$$

Erosion and dilation are dual operation which means that the erosion of an image A equals dilation of the image A complement. In other words erosion can be computed as dilatation of the image background and similarly for the dilation (eq. 2.30 and 2.31).

$$A \ominus B = \overline{\overline{A} \oplus H^*} \quad (2.30)$$

$$A \oplus B = \overline{\overline{A} \ominus H^*} \quad (2.31)$$

Due to the non-linear properties of the operations (they exist multiple input images with the same result after erosion), erosion and dilation are not inverse functions.

2.4.3 Opening

Opening is defined as dilation of the erosion of an image A by structuring element B (eq 2.32).

$$A \circ B = (A \ominus B) \oplus B \quad (2.32)$$

Opening is used for removal of small objects in the foreground (figure 2.4).

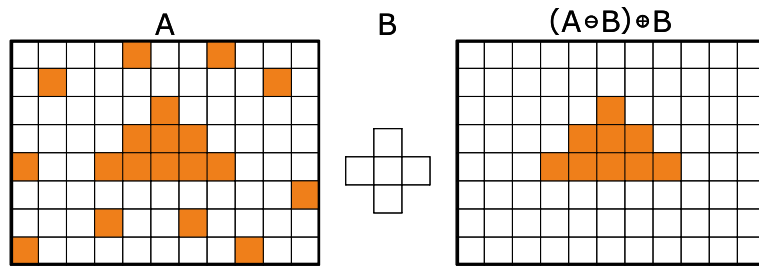


Figure 2.4: Opening

2.4.4 Closing

The closing of image A by B (figure 2.5) is defined as erosion of the image dilation (eq. 2.33).

$$A \bullet B = (A \oplus B) \ominus B \quad (2.33)$$

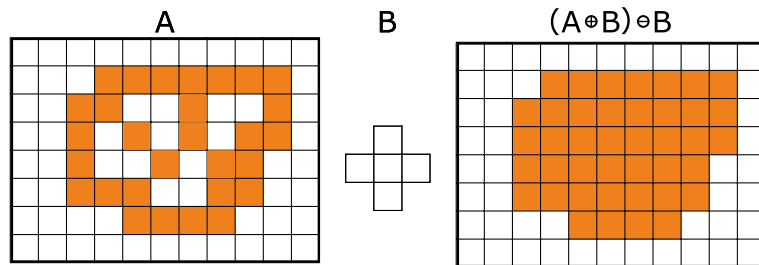


Figure 2.5: Closing

Closing is useful for closing small holes in the object. Together with opening is this method very effective for noise removal.

3 Image acquisition

Picture in digital world is a numeric representation of a two dimensional signal. The picture is stored in the computer memory as a combination of ones and zeros. Depending on whether the image resolution is fixed or not, we can determine the image type: vector or raster. Mostly the term “digital picture” defines a raster image.

Raster images have a finite set of digital values in rows and columns called pixels. Pixels are the smallest individual element in an image holding an information about the brightness of a given color at any specific point. To obtain these values and store them in our device a sensor in role of an interface between the digital and the real world is needed.

The two major types of digital image sensors used in camera systems are CCD (Charge-coupled Device) and CMOS (Complementary Metal Oxide Semiconductor) [11]. The main difference between the two types is in the technology and principle of acquiring and storing the image. CCDs are powered by a single amplifier and the information is collected from each diode at a time before uploading to camera memory. CMOS sensors have individual amplifiers and collectors for each photo diode which leads to a better energy and time efficiency [7]. CCDs are less prone to noise and more sensitive to light than CMOS sensors resulting in better brightness in dark situations. In other words, pixels in CCD sensors are committed to capturing light which produces higher-quality images but it can causes the camera to be less efficient. CMOS sensors cut out a step by integrating technology into the chip that converts visual information to digital data as it is processed to the chip. This makes the sensors highly efficient although the extra task of the pixels can result in increased noise in the final image [16].

The resolution of a digital camera is limited by the image sensor and by the used camera lens [14]. The number of pixels in the sensor determines the camera’s “pixel count”. The pixel count in a typical sensor is calculated as product of the row count

and the columns count. For example a 1,000 by 1,000 pixel sensor has 1,000,000 pixels (1Mpx).

After the process of taking and saving the picture in the internal memory a way how to save the picture to corresponding device. The transfer of the data must be dealt with from both SW and HW point of view. The most used hardware is USB or FireWire interface and the software depends on the type of the camera.

3.1 Hardware

3.1.1 Camera

Camera is in image processing a key element. The resolution and the noise of the sensor affects the processed image. In the best case scenario it influences only the time needed to process the image, in the worst case scenario the image becomes unprocessable. Two types of cameras were used for testing: a webcam and two industrial uEye cameras.

uEye

IDS uEye is an industrial camera often used for image processing methods with USB interface and resolution from WVGA up to 10 Megapixel. The used model is a UI - 2230 with a resolution of 1024 horizontal pixels and 768 vertical pixels (generating the resulting matrix with 786 432 points in each channel).

Maximal frame rate of such device is 74 fps in AIO mode with a resolution of 256x256 but for the use of crack capturing the use of 30 fps is sufficient. The exposure time can be set from 0.066 ms to 1040 ms. This camera has a 12-bit ADC which means that for each color value we have 4096 possibilities for each color. Those cameras have a C-mount connection which makes them more universal than the standard web cameras for PCs. Such device allows a better possibility for adjustment as they can be used with a lens and filter system. The main drawback is its price and the limited possibility of use without the original capturing uEye Cockpit program. The main reason for the incompatibility between the uEye and other programs is that these cameras are listed as a multifunction HW and not as a capturing device. This fact is given by the drivers distributed by the IDS Company. One of the advantages is the possibility to communicate in both ways (Camera-PC, PC-Camera) so the settings of the device can be changed directly. IDS uEye cameras

have very sophisticated control flow with almost unlimited possibilities discussed in section 3.2.2.

Webcam

Webcam is much simpler device than an industrial camera, it has a CMOS or a CCD sensor and no mount for a lens system. These devices have mostly only a one way communication possibility (they stream video and audio stream). The received data can be then processed and stored. Most settings like contrast and brightness of these cameras are changed on the received image with algorithm rather than on the CMOS/CCD chip settings. Webcam have also a smaller resolution than the industrial cameras but they are easier to use.

3.1.2 Camera lens

A lens is an optical device affecting the focus of a light beam through refraction. In photography are lenses part of a camera body and together with other camera mechanism allow to make images of objects. The lenses used in camera, video camera or e.g. microscope do not differ in principle but they have different construction.

IDS uEye cameras dispose of C-mount which allows the camera lens system to be mounted. In this case there is no electronic control for the mounted setup thus are the lenses controlled manually. Three main parameters determines lenses properties: focal length, maximum aperture and minimum focus. The used lenses has a fixed focal length with adjustable aperture and focus.

The distance between the lens and the image sensor with the object in the focus is called the focal length of the lens. The value is usually given in millimetres with typical values of 28 mm, 50 mm or 100 mm. For the zoom lenses are both maximum and minimum focal length stated (e.g. 18 mm- 50 mm).

An aperture is an opening in which the light can travel. It has the possibility to control the amount of the light reaching the film or sensor. The lens specification includes maximum and minimum aperture. The lenses with higher maximum aperture (lower f-number) create a brighter image. Moreover wider maximum aperture allow smaller depths of field that soften backgrounds. This creates great softening image and the object appears separated from the background.

The minimum focus distance is the shortest distance at which a lens can focus. If the distance between the camera and the photographed object is smaller the acquired image is blurred.

3.1.3 Filter

Term filter in context of photography and videography represents a camera accessory with an optical filter which can be inserted into the optical path. It is used for adjusting the exposure and bringing out colours and textures. In most cases is the filter mounted on the camera lens however on the test stand is the filter mounted on the adjustable light source.

Filters have a high influence on the process of detection cracks (figure 3.3 - 3.11). Experimentally was found out that filters containing orange colour have the best properties for accentuation of the cracks on dark colored background. There are several types of filters, for example color filters, polarization filters, clear filters and UV filters. Each type has specific function and combination of more filters is in some cases possible. Color filters work on subtraction of certain colors of light, letting the remaining colors through. Such filters can be used for contrast enhancement, mostly in black and white photography. For example yellow, orange and red filters can be used for enhancing the contrast between the clouds and the sky. This effect is getting stronger as the intensity of the red color in the filter increases. Polarizing filters are colorless therefore they do not affect the color balance. Rather they filter out light with specific direction of polarization.

There are properties of four used filters in figure 3.2. In the tables are X,Y chromaticity co-ordinates according to CIE 1931. The absorption (abs) of a filter is calculated from the Y% value. It brings another way of expressing the light-stopping properties of that filter. Thanks to the linear properties of abs, the values can be added or subtracted more easily than using Y%. The graph showing transmitted light (Y%) is representative of overall average transmission of that filter, as perceived by the human eye. The (Y%) value is one of the Tristimulus values (a set of values unique to each colour) that are calculated mathematically from the data contained in the SPD (Spectral Power Distribution) graph.

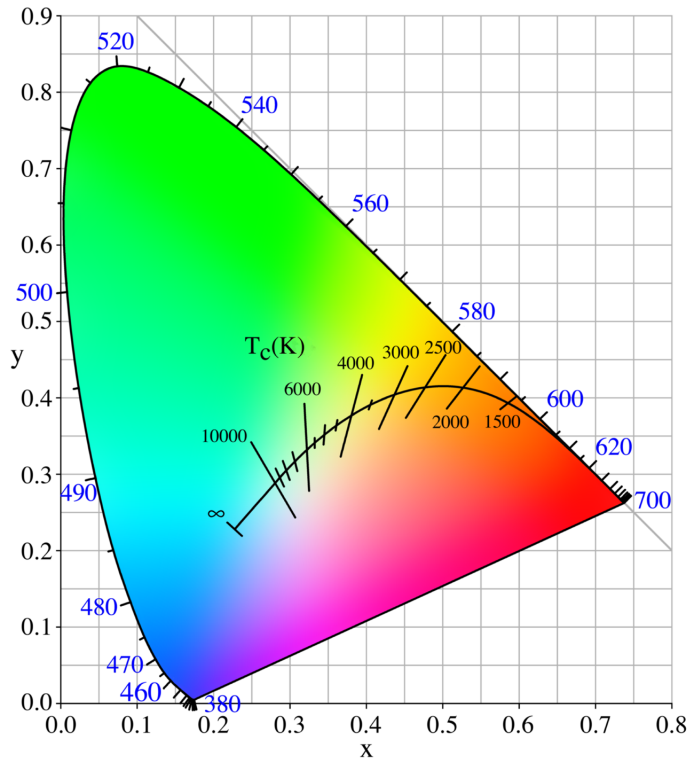
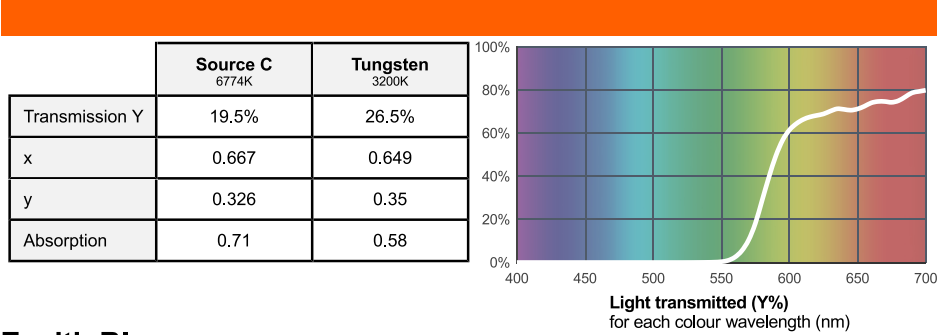


Figure 3.1: CIE 1931 diagram

Deep Golden Amber



Zenith Blue

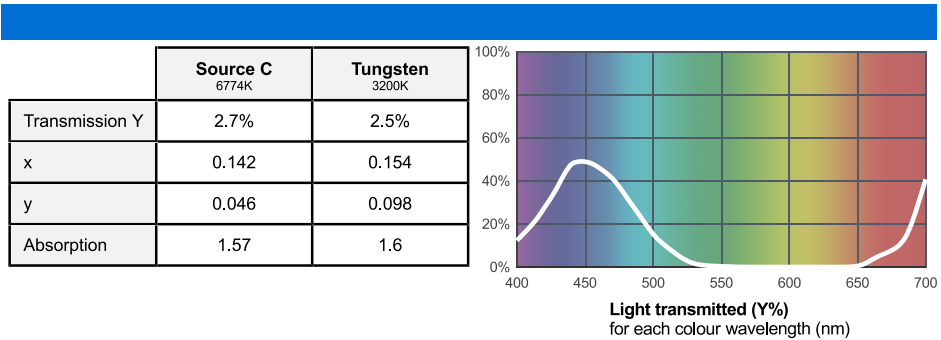


Figure 3.2: Properties of the used filters

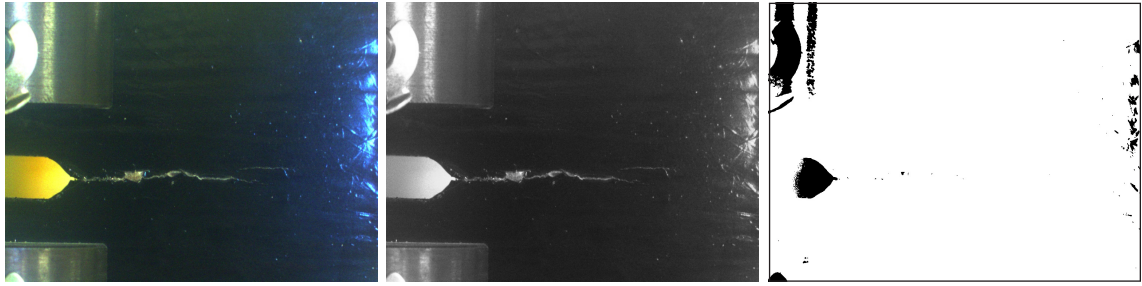


Figure 3.3: Without a filter

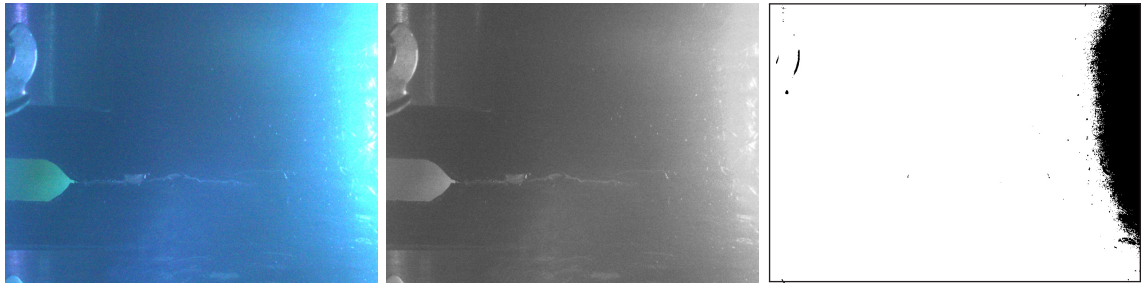


Figure 3.4: Blue filter with high light intensity - Zenith blue

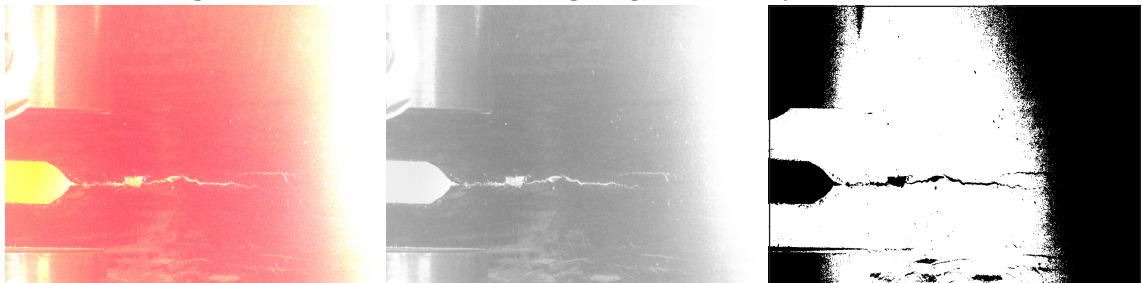


Figure 3.5: Orange filter with high light intensity

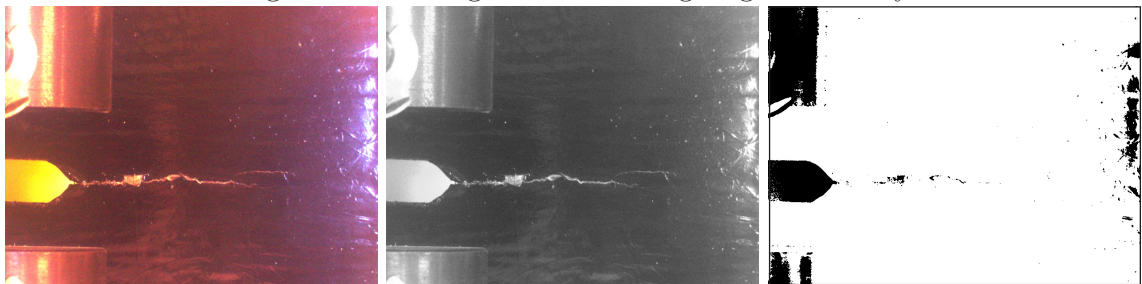


Figure 3.6: Orange filter with low light intensity

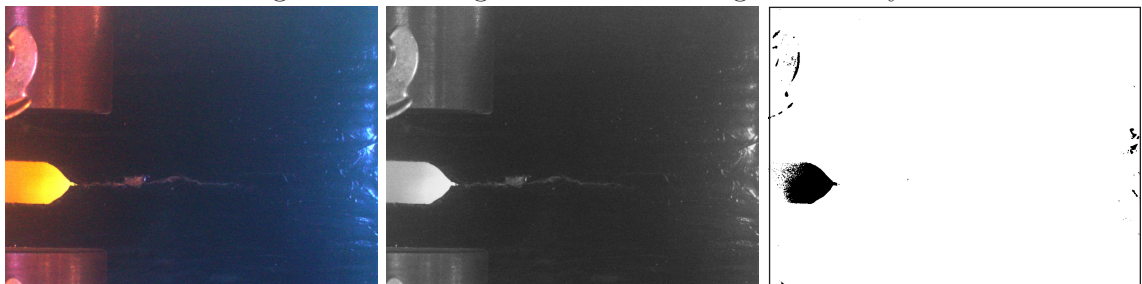


Figure 3.7: Combination of blue and orange filter with low light intensity

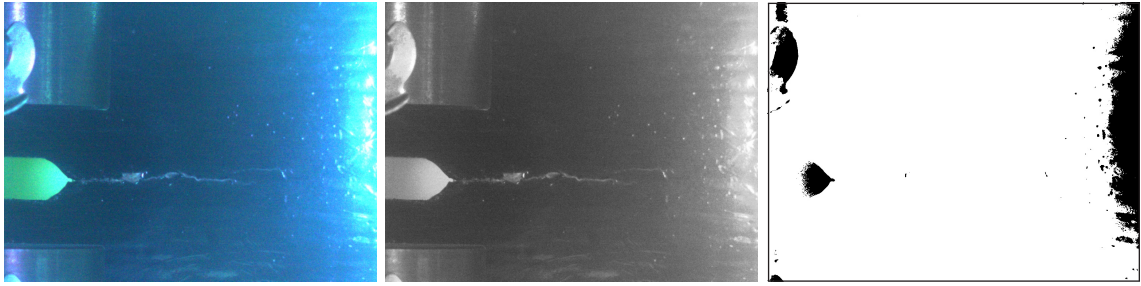


Figure 3.8: Blue filter with high light intensity - Regal blue

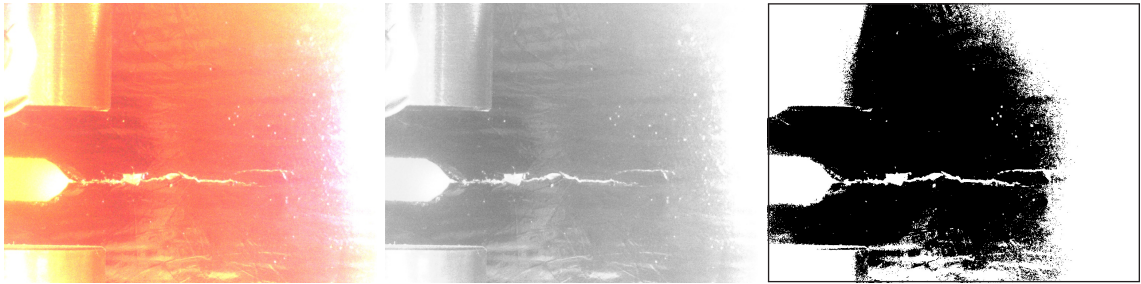


Figure 3.9: Orange filter with high light intensity - Cool LED Orange

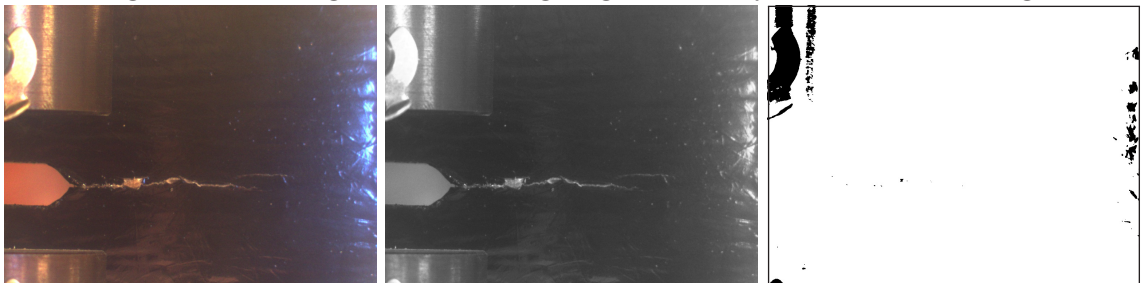


Figure 3.10: Orange filter with low light intensity - Cool LED Orange

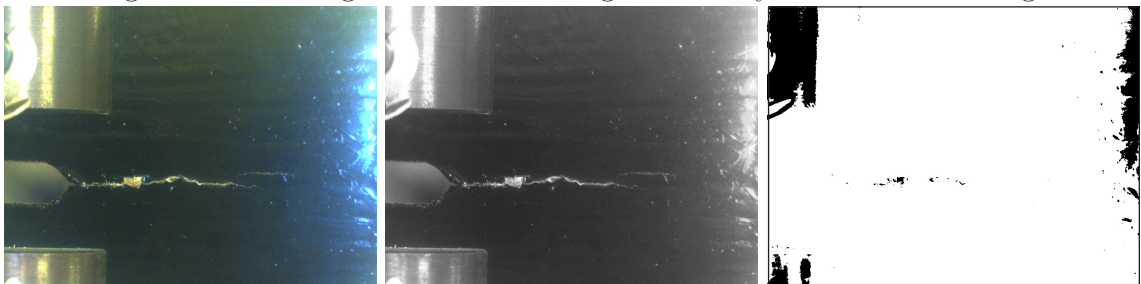


Figure 3.11: Green filter with low light intensity

3.2 Software

Microsoft[®] Windows has at its disposal an integrated program for capturing videos and images from standard plug and play cameras. This integrated program has very limited possibilities but in some applications is sufficient for capturing and archiving images from the cameras. However, this program has two major drawbacks. First drawback is that it can use only one camera at a time and the second drawback is that it can use only cameras which are listed under windows as capturing devices. As mentioned in section 3.1.1, the industrial cameras are seen by computer as not specified devices and hence they cannot be used with this program. Industrial IDS cameras can be used only with special drivers and program, which can be obtained from the manufacturers websites. Unfortunately, this program can only work with IDS camera systems. Meaning that for the capturing with two different cameras, two running programs, two computers memory access processes and two communication buses would have to be used at the same time. For the use of any combination of uEye cameras and webcams there is a need to use an own program. For the use of webcams it is possible to use standard OpenCV 3.0.0 libraries, but for the use of the uEye cameras we will need to use external libraries and write our own methods.

3.2.1 WebCam

The use of WebCam capturing device is simple thanks to the created framework libraries. Concept of such framework is explained with the help of a very simple acquisition program (code 3.1) which is also supposed to demonstrate the philosophy of OpenCV and use of the pre-defined functions.

Code 3.1: Aquisition program for WebCam

```
VideoCapture cap(0); // cv::VideoCapture is a class for video
    capturing, cap() is the name and 0 is the number of the first
    installed camera
if(!cap.isOpened()) // Check if we succeeded in opening the port to
    the camera, otherwise terminate the program with error code
    (return -1)
    return -1;
namedWindow("MyWindow_nrX",1); // Create a window with parameters:
    namedWindow(const char* name, int flags=CV_WINDOW_AUTOSIZE), name
    is identifier that will windows use for the window caption,
    supported flags are: WINDOW_NORMAL, WINDOW_AUTOSIZE or
    WINDOW_OPENGL
for(;;)
{
    Mat frame; // Create a matrix variable
    cap >> frame; // Get a new frame from camera and save it to
        frame
    cvtColor(frame, frame, CV_BGR2GRAY); // Converts an image from
        one color space to another. The syntax is cvtColot(source,
        destination, code of color space)
    imwrite("../images/Gray_Image.jpg", frame); // Saves an
        image to disk with a syntax imwrite("PATH/name.format",
        source);
    imshow("Image", frame); // Shows saved image in our Mat on the
        monitor. Syntax is imshow("name", source);
    if(waitKey(30) >= 0) break;
}
```

As presented in the code above, the main idea resides in opening the device which is listed in windows devices with assigned number. For example in a notebook with an integrated camera and another web camera, the integrated camera has number zero and the external camera has number one. Special cases are uEye and another industrial cameras which are also counted from zero. In the practical application it means that in the notebook with integrated web camera and IDS/uEye camera connected, both have number zero but they are listed in different device class.

3.2.2 uEye

As mentioned before uEye requires a specific control procedure. The dynamic libraries included in the original driver can be used but first it is necessary to understand them and create methods for their use. The information about the camera is stored in CAMINFO register inside the camera EEPROM. It can be accessed by use of function `is_GetCameraInfo()`. This information is then read out in form of 64 byte data structure (table 3.1).

Table 3.1: Structure containing data about camera

| Type | Register name | Register content |
|---------------|---------------|--|
| Char | SerNo[12] | Serial number of the camera |
| Char | ID[20] | e.g. "IDS GmbH" |
| Char | Version[10] | "v1.00" or later |
| Char | Date[12] | "01.08.2004", date of quality check |
| Unsigned char | Select | Camera ID |
| Unsigned char | Type | Camera type (64 = uEye USB 2.0 14 bit) |
| Char | Reserved[8] | Reserved |

With the information it is possible to access the camera memory and set the desired parameters. Please find examples of the functions used in the main algorithm in the list below. For all the function please refer to the attached CD (appendix A).

- **Color formats:** something
- **Possible image output:** something

The next step is to access the camera, make the initialization steps and read the image.

Initialization steps

- **is_InitCamera()** – Opens the driver and establishes contact with the HW. Without this instruction the camera will not be connected to the system.
- **is_ExitCamera()** – Closes the camera and deallocates memory which was allocated by the SDK (Software Development Kit).

Selection of operating modes and return of properties (camera list, brightness, contrast, color corrections, output image format, color depth etc.):

is_GetCameraList()

With this information about the attached cameras can be queried. In order to obtain all information, the field size must be adapted to the number of connected cameras. In the following table 3.2 the used structures are described.

Table 3.2: Used structures

| Type | Register name | Register content |
|------------------|---------------|---|
| Ulong | dwCount | Number of cameras connected to the system |
| UEYE_CAMERA_INFO | uci[1] | Structure of the uEye CAM info [20] |

- **is_SetColorMode()** sets the required color mode in which is the image data saved or displayed by the VGA board. When saving the image data it is important to allocate enough memory (depending on selected color mode). For possible parameters please refer to the datasheet [20]).
- **is_AllocImageMem()** allocates image memory for an image with given height, width and color depth according to the equation 3.1.

$$\text{size} = \left[\text{width} \cdot \left(\frac{\text{bitapixel} + 1}{8} \right) + \text{adjust} \right] \cdot \text{height} \quad (3.1)$$

Calculation of line increments is depicted in equations 3.2 and 3.3.

$$\text{line} = \text{width} \cdot \left[\frac{(\text{bitapixel} + 1)}{8} \right] \quad (3.2)$$

$$\text{line.increment} = \text{line} + \text{adjust} \quad (3.3)$$

with

$$\text{adjust} = 0 \quad \text{if line is divisible by 4 without rest}$$

$$\text{adjust} = 4 - \text{line}\%4 \quad \text{if line is not divisible by 4 without rest}$$

- **is_SetImageMem()** activates the allocated image memory for receiving the image.

- **is_FreezeVideo()** digitizes an image and transfers it to the active image memory. In DirectDraw mode the image is digitized in the DirectDraw buffer (either on the VGA card or in a back buffer). If you are using ring buffering, the image is recorded to the next non-locked image buffer in the sequence. As soon as the last non-locked sequence buffer has been filled, the sequence event/the sequence message is triggered. The picture recording takes place triggered, if the trigger mode were activated before.

Image acquisition and memory management:

- **is_GetActiveImageMem()** returns the pointer to the beginning and the ID number of the active memory.

The functions described above give an sense of the program structure but they cover only a very small part of functions used in the program. Due to the complexity of the program are the used functions not further discussed and the program run is represented with flow graphs (the functions details can be found in [18]). The flow graph shown on figure 3.12 depicts algorithm used for capturing images with WebCAM. The figure 3.13 represents the flow diagram of algorithm dedicated for uEye.

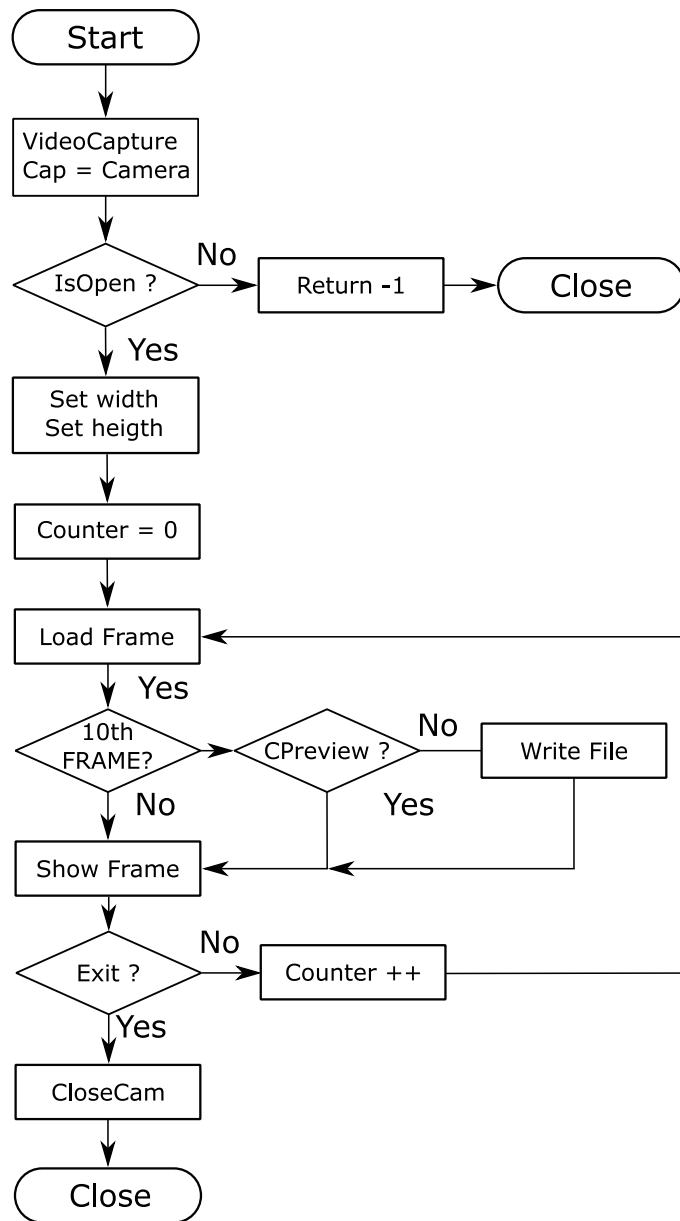


Figure 3.12: WebCAM flowchart of used algorithm

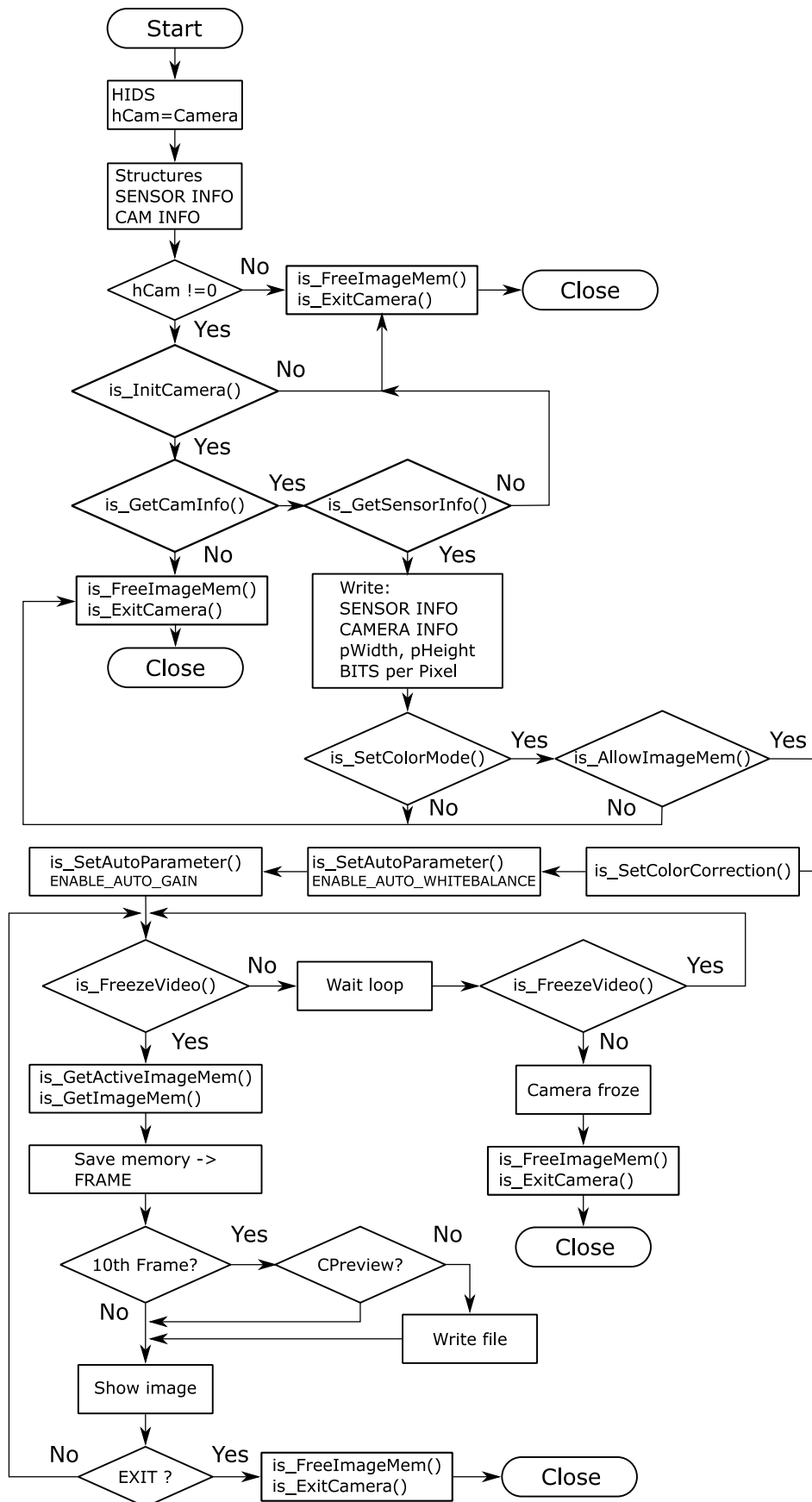


Figure 3.13: uEye flowchart of used algorithm

3.2.3 Final program

The final program PictureMaker includes algorithms for use of both WebCam and uEye camera (figure 3.16). After the program is started the window for setting up the parameters appears (figure 3.14).

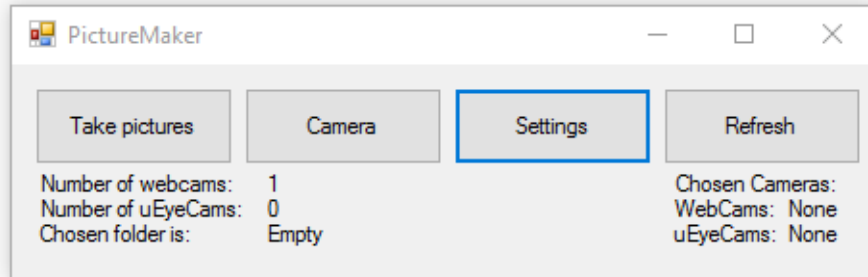


Figure 3.14: Capturing program

The window contains four buttons and information about connected devices. The functions of the buttons are described in the following list.

- **Take pictures**

The destination folder is opened and the check for older images from the cameras is initialized. After that one or two windows (depending on number of connected devices) are opened with camera previews and every tenth frame saved.

- **Camera**

This button opens a setting window in which the parameters of the test stand can be adjusted. The cameras preview opens but no pictures are saved.

- **Settings**

A new dialogue window with possibility of changing the camera and destination folder setting opens (figure 3.15).

- **Refresh**

The information on the main window are actualized.

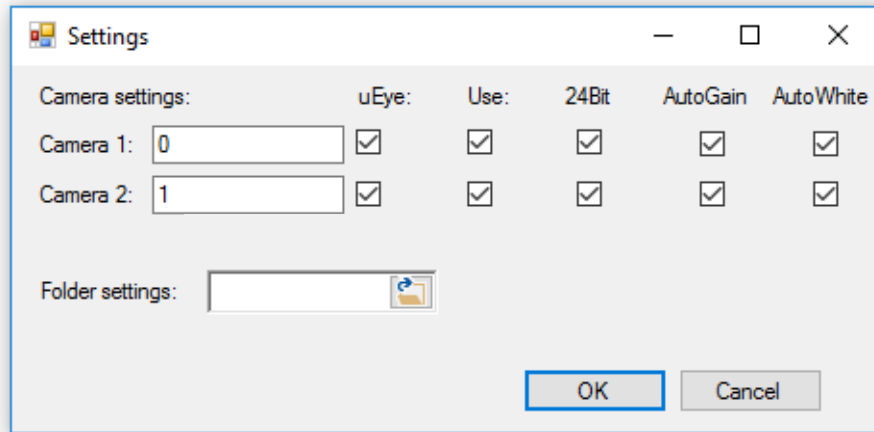


Figure 3.15: Capturing program settings

Functions *Camera* and *Take picture* are called with the parameters set in *Settings*. Each call is made by an interrupt and passes to the given function the information about type of the camera, number of the camera and target folder for saving the pictures.

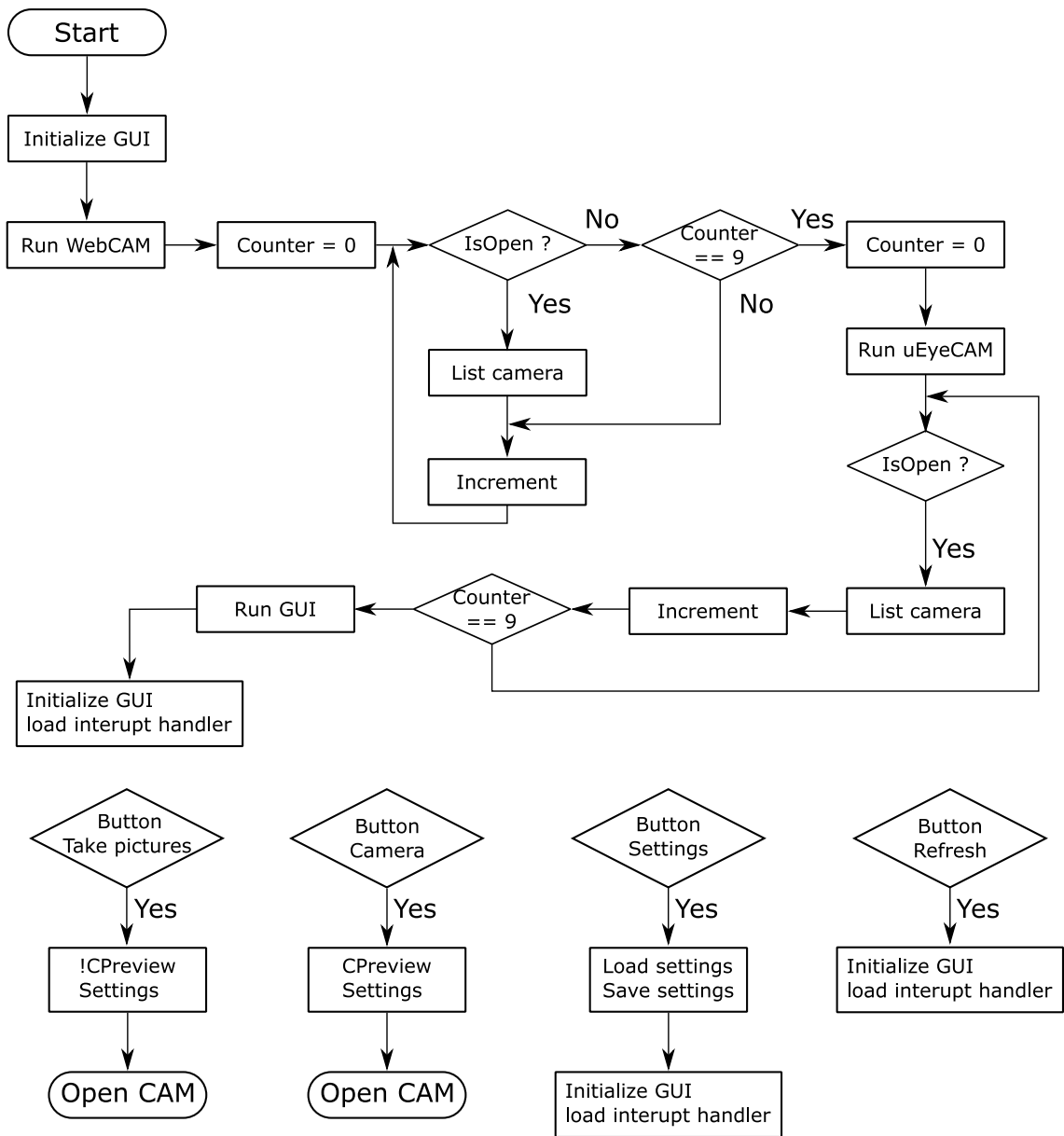


Figure 3.16: Simplified capture program flow diagram

4 Image processing

4.1 OpenCV 3.0.0

OpenCV¹ (Open Source Computer Vision Library) is an open source computer vision and machine learning software library released under a BSD license. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products [17].

The roots of those libraries are based on Intels INDE (Integrated Native Developer Experience) OpenCV. Intel[®] INDE OpenCV IDE (Integrated Development Environment) integration provided a framework for the use under the Microsoft VS (Visual Studio), Eclipse and Android Studio. Standard optimization with OpenCL[™] and Intel[®] IPP (Integrated Performance Primitives) and Intel[®] TBB (Threading Building Blocks) ensured Intels INDE OpenCV applications optimization to Intel's heterogeneous platforms, accelerated with Intel[®] HD and Intel[®] Iris[™] graphics family. Now these features are available under the above described OpenCV.

The library has over 2500 optimized algorithms, including a set of both classic and state-of-the-art computer vision and machine learning algorithms. This library is used by well-established companies like Google, Honda, Sony, or Microsoft. It was even used as a detection of intrusions in surveillance videos in Israel or for monitoring of mine equipment in China ².

OpenCV was designed for computational efficiency and with a strong focus on real-time applications on every OS. For that reason it takes advantage of MMX and SSE instructions. A program written in optimized C/C++, with a use of these libraries can take advantage of multi-core processing or enabled with OpenCL, it can take advantage of the hardware acceleration.

Last versions of OpenCV are able to use the hardware support of CUDA (Compute Unified Device Architecture). CUDA³ is a parallel computing platform and

¹<https://developer.nvidia.com/opencv>

²<http://opencv.org/about.html>

³http://www.nvidia.com/object/cuda_home_new.html

programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the GPU. The possibility to use a CUDA support enabled the use of OpenCV in medical, chemical and aerospace research. For example the GPUs can simulate blood flow and identify hidden arterial plaque without invasive imaging techniques or exploratory surgery. According to NASA [21] the use of a CUDA model reduced the computational time from ten minutes to three seconds.

4.2 Pre-processing

The main task of pre-processing is to prepare the acquired image for the processing part, meaning to accentuate the real crack and reduce the influence of the surrounding objects created by material, light and camera properties. This can be done by several methods. Two of the used methods are described. The first method which was used and developed is based on standard thresholding algorithms. It takes an advantage of grey-scale images. The uEye cameras have the possibility to record and store monochromatic images. However, the cameras do not produce grey-scale images by direct acquisition but rather by conversion of the taken image from RGB into 8 bit grey-scale image. That means that this conversion can be done in computer as well and both RGB and Grey-scale images are available.

The second method is based on edge and corner detection method. The detection of corners using a morphological corners is not implemented in OpenCV 3.0.0 and there is a need for custom definitions of non-square structuring elements. The main advantage of such algorithm is its independence on the threshold value.

4.2.1 Reading of a scaling factor

To be able to analyse the crack length it is necessary to know the size of the pixel in the test probe image in millimetres. The scaling is possible only if there is a reference. In this particular case is the millimetre paper placed next to the test probe (figure 4.1).

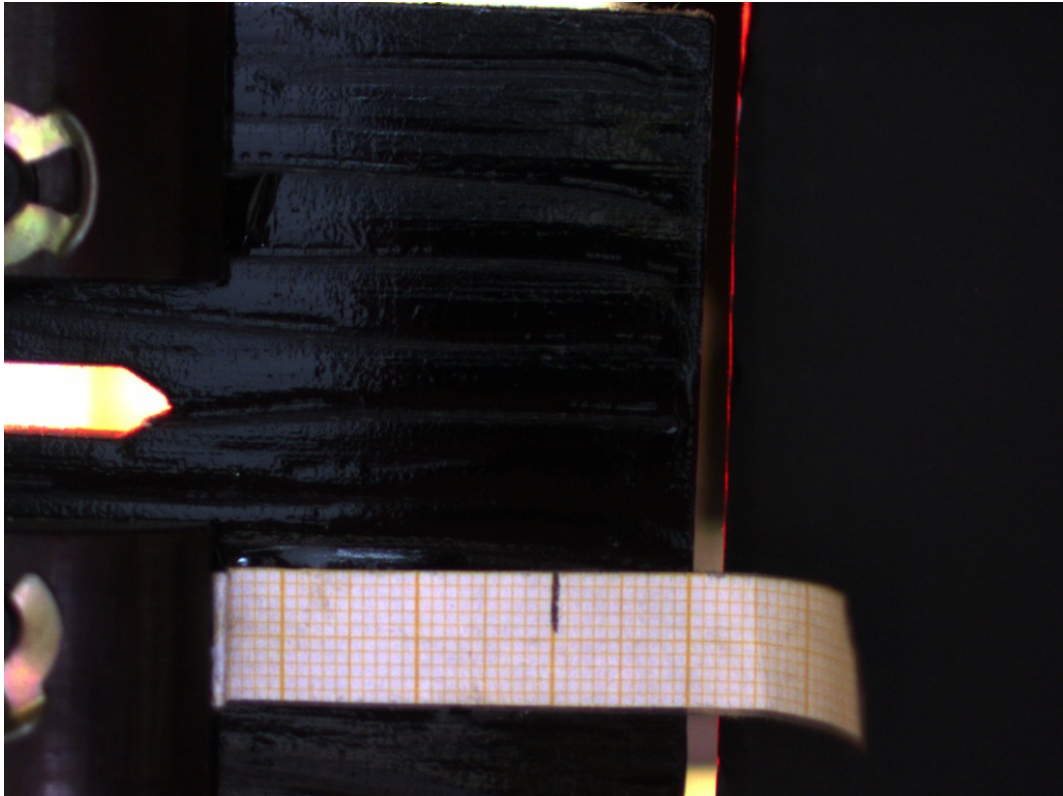


Figure 4.1: Image for taking the scaling factor

The millimetre paper is not placed on the same place in every picture and therefore it is first necessary to find its position. The area with the millimetre paper is found with use of template matching algorithm. Then it converted into grey-scale and an adaptive threshold filter is applied (figure 4.2).

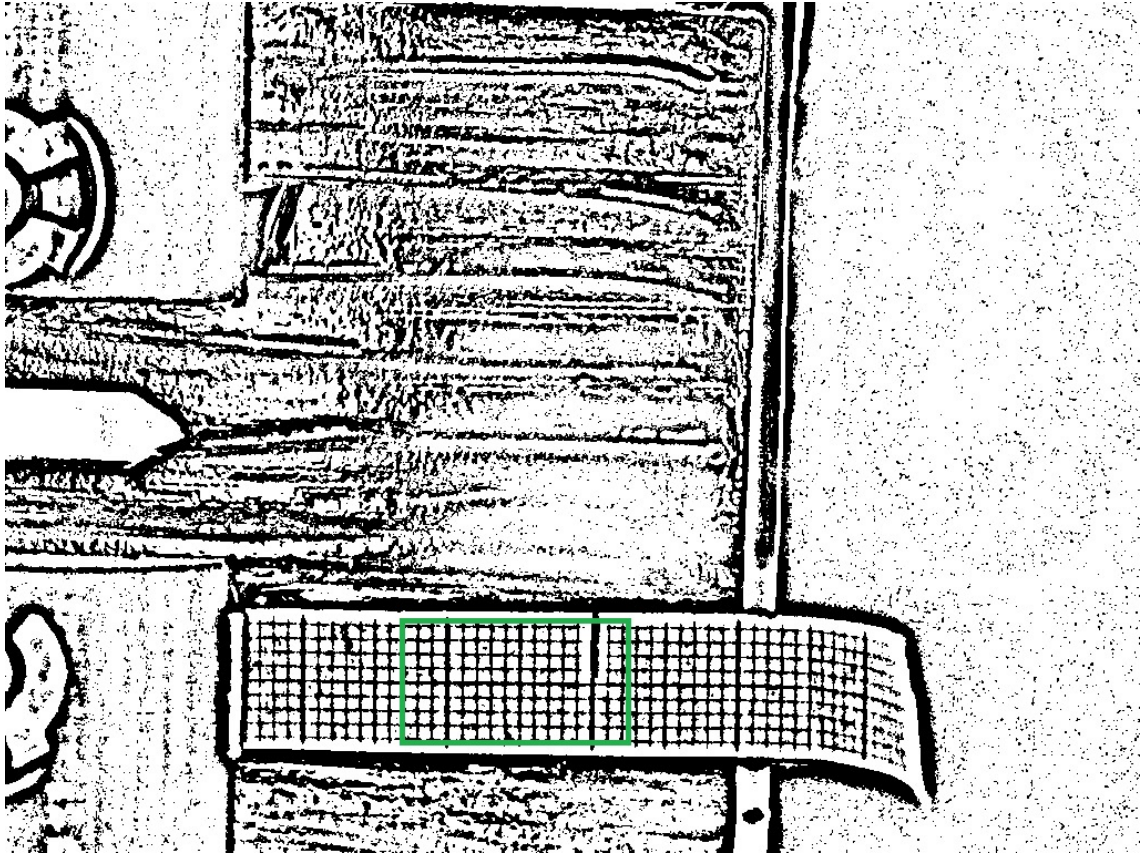


Figure 4.2: Thresholded image with template matching

Such data are stored in a matrix (two dimensional array) and the mean value between the lines is calculated as a mean value in x and y direction (eq. 4.1).

$$length_{xy}[mm] = \frac{\text{Number of white pixels}_{xy}}{\text{Number of black pixels}_{xy}} \quad (4.1)$$

With use of the previous equation is the scaling factor for the crack analysis obtained.

4.2.2 Thresholding method algorithm

The first algorithm is based on classical thresholding methods and generates sufficient results for the first set of test probes. After installing the uEye camera the results became insufficient due to higher resolution and better brightness. The screen from the uEye camera (figure 4.3) is used for comparison of both algorithms.

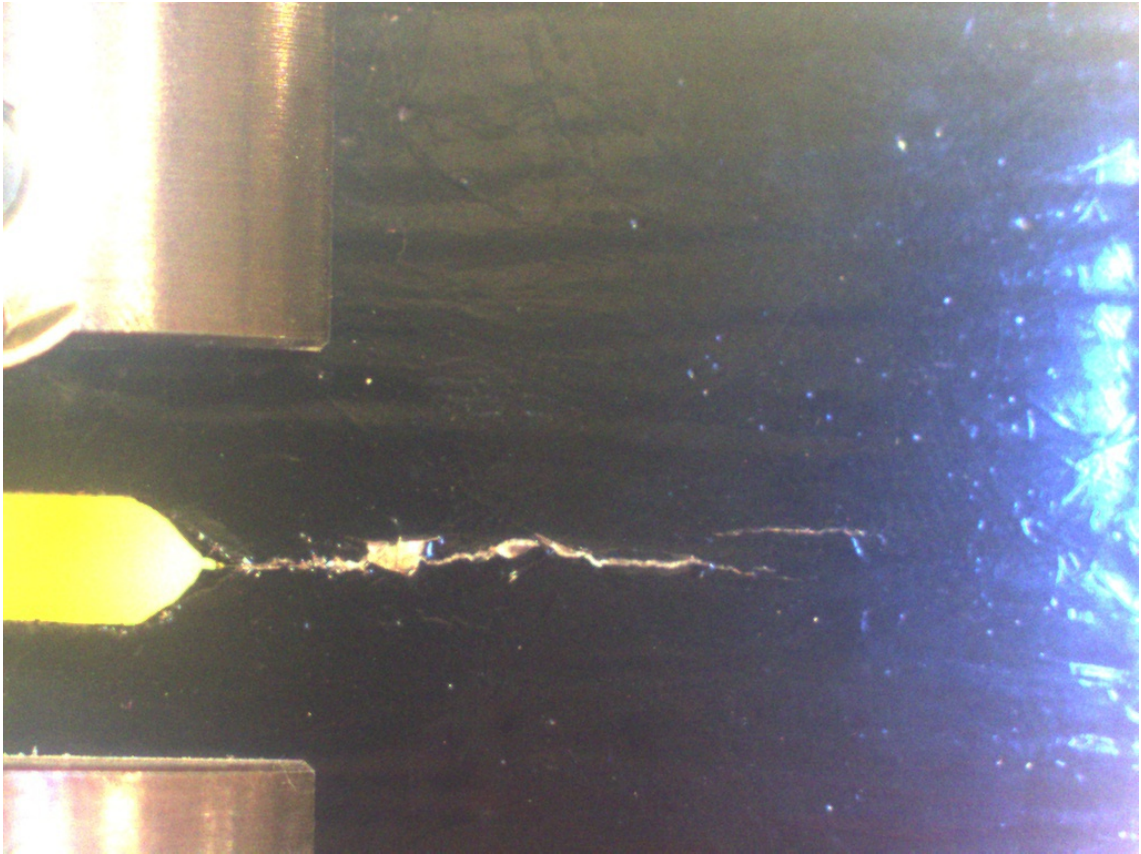


Figure 4.3: Input data from camera

In the first step is the image converted into grey-scale image. After this conversion two thresholding methods are used. First method is the Otsu's method mentioned in section 2.2.1 (figure 4.4). The second method is thresholding with use of the Gauss method (figure 4.5). In both results is the crack visible and at least partially separated from the background. These crack pictures can be modified with use of the contrast stretching method. For the contrast stretching it can use a standard procedure of calculation a linear function (equation 4.2).

$$g(i, j) = \alpha \cdot f(i, j) + \beta \quad (4.2)$$

Where α is called gain and β is called bias.

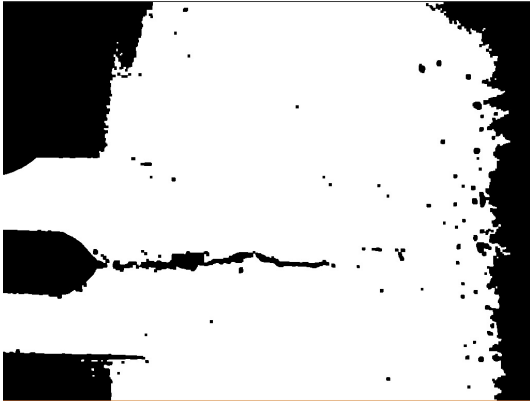


Figure 4.4: Treshold Otsu

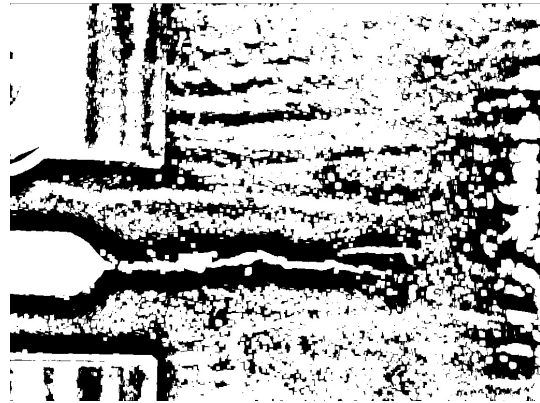


Figure 4.5: Threshold Gauss

The method using the Otsu's algorithm is better but it shows only a part of the crack and therefore it cannot be used without any modifications. The method using the adaptive mean Gauss algorithm shows the whole crack, but the length cannot be estimated or calculated due to the light influence at the end of the probe. In some cases it is possible to use such preprocessed data but it is not recommended for the real analysis of the crack length.

Both methods were implemented into the final program but after the change of the cameras they could no longer be used for the crack path analysis but they are still sufficient for the crack path estimation. Because of the swiftness of such processing they are a useful indicator of the growth direction of the crack and therefore both algorithms (algorithm using morphological method and algorithm using thresholding method) are used in the preprocessing.

4.2.3 Morphological method algorithm

Load an image and create kernels

First operation is loading an input image. For a better comparison is used the same image as in the previous method (figure 4.7). Then two kernels are created for use in morphological operations (described in section 2.4). These kernels are cross and diamond (figure 4.6).

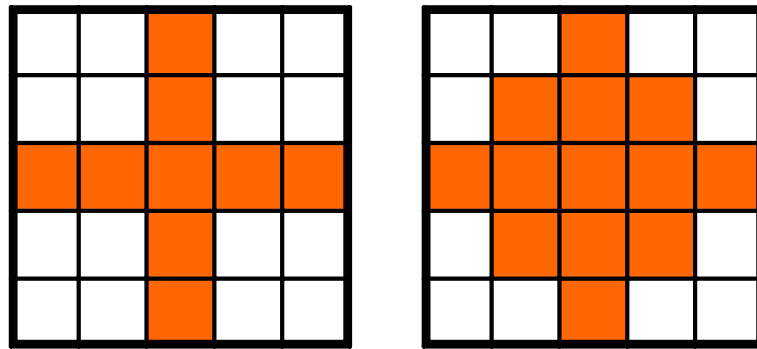


Figure 4.6: Cross (left) and Diamond (right) kernel

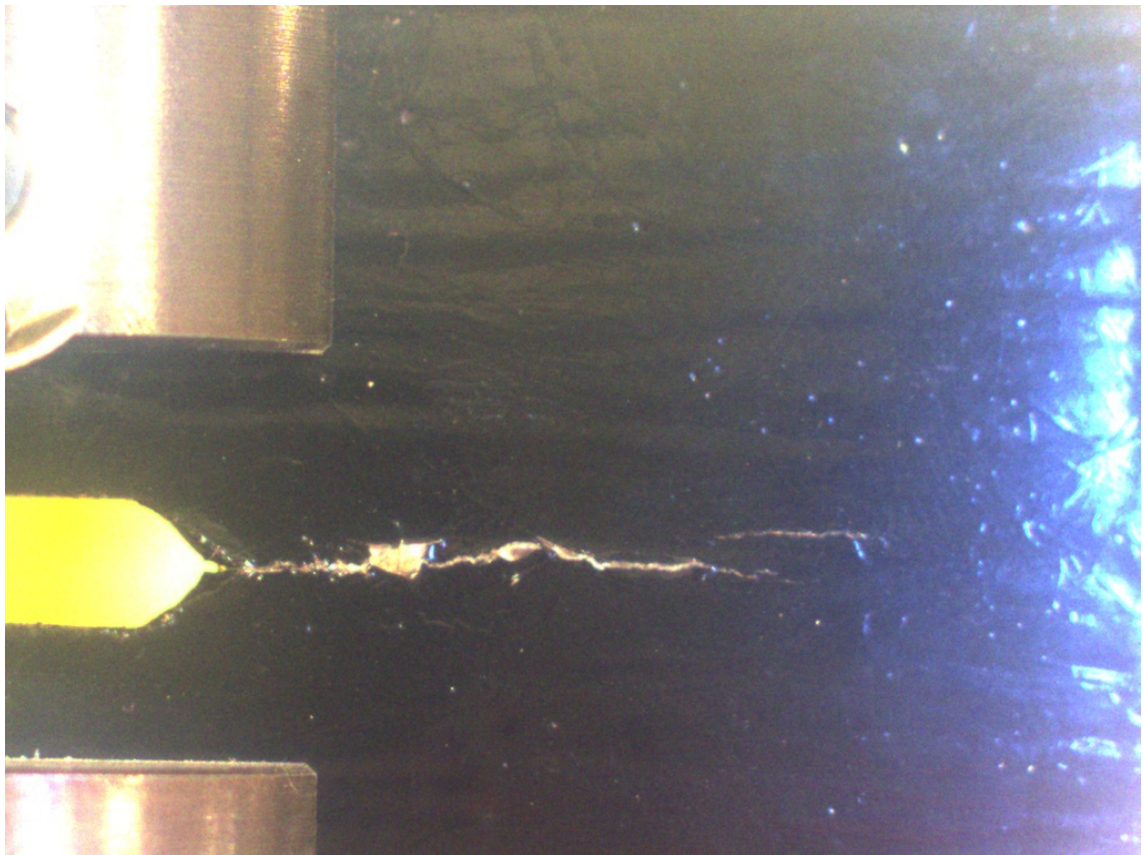


Figure 4.7: Input data from camera

Morphological gradient

First the kernels are created and the image for processing is loaded. A matrix for holding of the result is created and filled with a image obtained by using the morphological gradient. The morphological gradient is calculated as follows (eq. 4.3):

$$\begin{aligned} \text{gradient}(\text{src}, \text{element}) &= \text{dilate}(\text{src}, \text{element}) - \text{erode}(\text{src}, \text{element}) \\ &(\text{element is the standard plus kernel}) \end{aligned} \quad (4.3)$$

The resulting image after applying the function is depicted on figure 4.8. The gradient operation separates the edges of the crack by the means of whiter color.

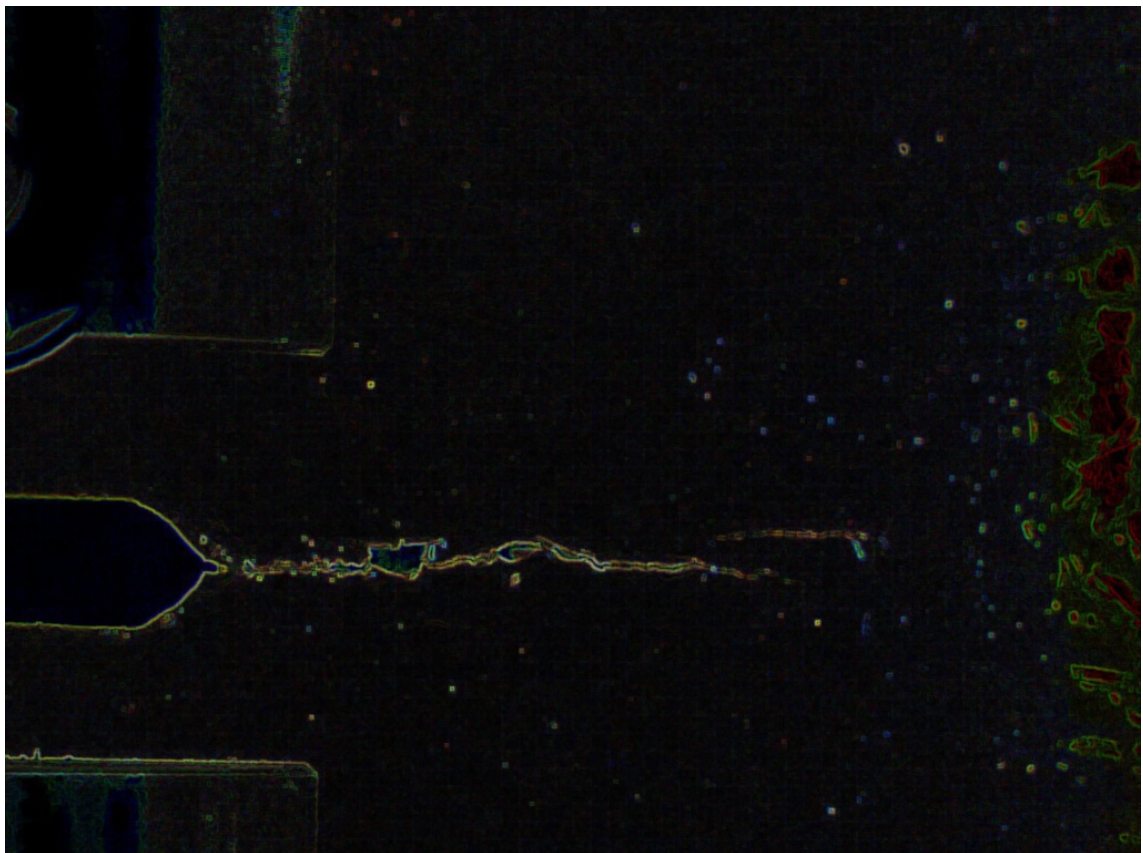


Figure 4.8: Gradient of the image

Erode with diamond kernel

Edges correspond to a rapid transition between darker and brighter pixels. An erosion operator is applied on such difference. The net result replaces each pixel by the lowest value in a certain neighbourhood, thus reducing its height (value in 3D graphic). As a result, such peaks “erode” as the valleys expand.

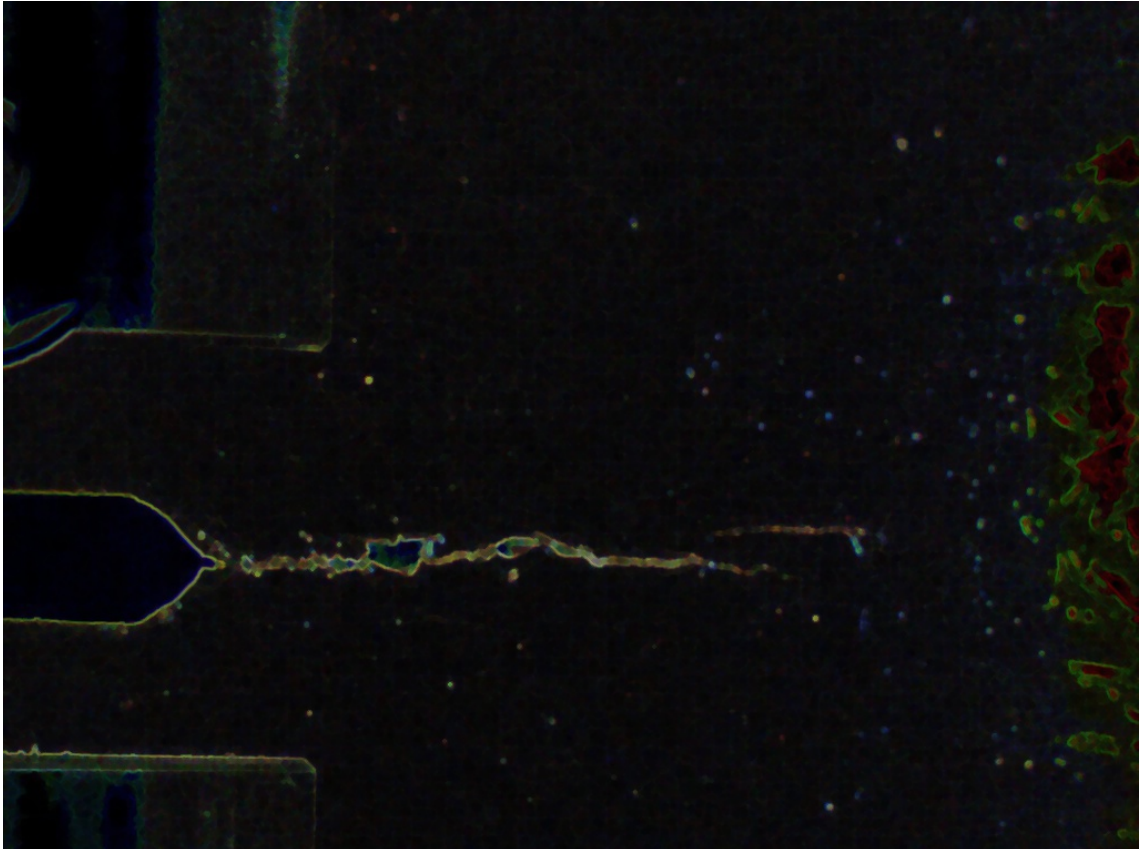


Figure 4.9: Erosion

Dilate with cross kernel

Dilation has the exact opposite effect, peaks grow in width and the valleys are narrowed. However, in both cases the area of constant intensity remains relatively unchanged.

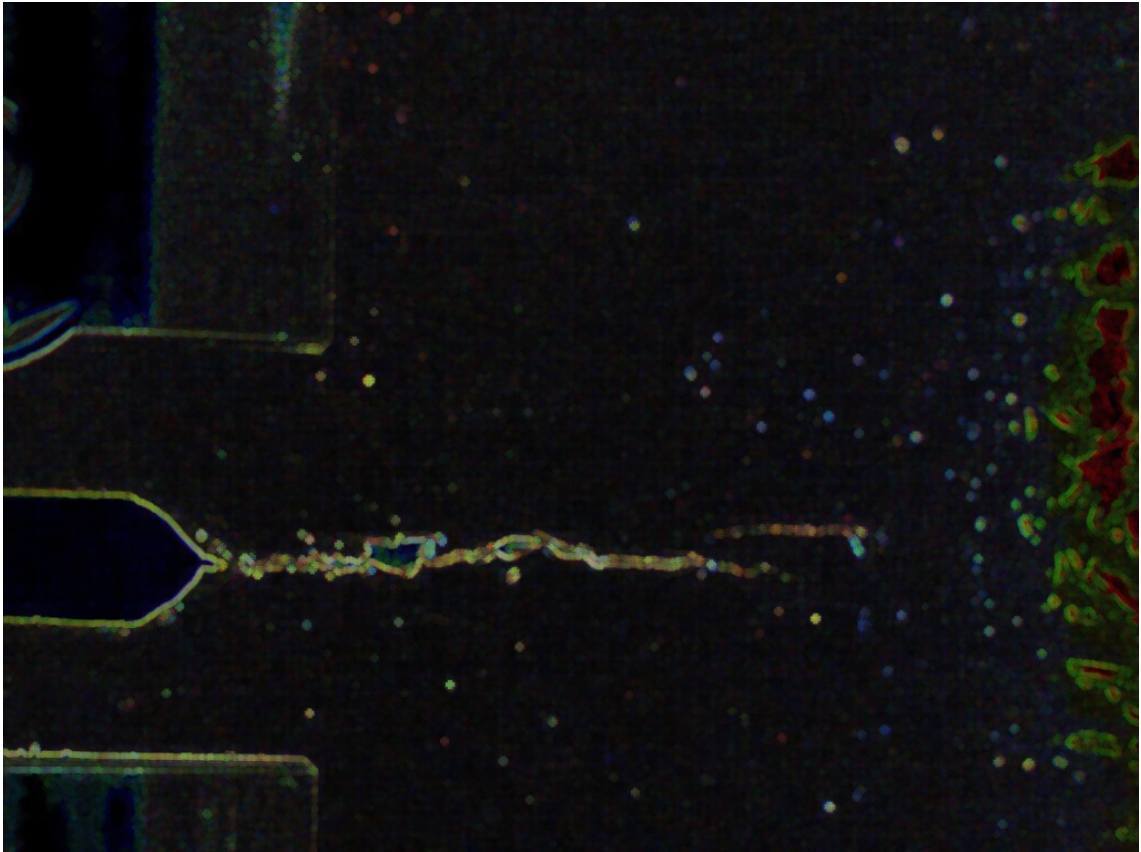


Figure 4.10: Dilatation

Using contrast stretching and saturation

For the contrast stretching can be again used a standard procedure of calculation a linear function with a slight modification (eq. 4.4).

$$g(i, j)[channel] = \alpha \cdot f(i, j)[channel] + \beta \quad (4.4)$$

Where α is called gain, β is called bias and channel is the number of RGB channel. Sometimes these parameters are said to control contrast and brightness respectively.

```
Result.at<Vec3b>(y, x)[c] =  
    saturate_cast<uchar>(alpha*(MorfoRes.at<Vec3b>(y, x)[c]) + beta);
```

Saturated_cast makes sure that the resulting values are valid. Each pixel and color separately thus a triple loop is used:

```
for (int y = 0; y < result.rows; y++)  
{  
    for (int x = 0; x < result.cols; x++)  
    {  
        for (int c = 0; c < 3; c++)  
            { /*Function for saturation*/ }}}
```

This method allows the use of saturation in each RGB channel, in case of use a filter, we can set up the parameters directly for that color channel.

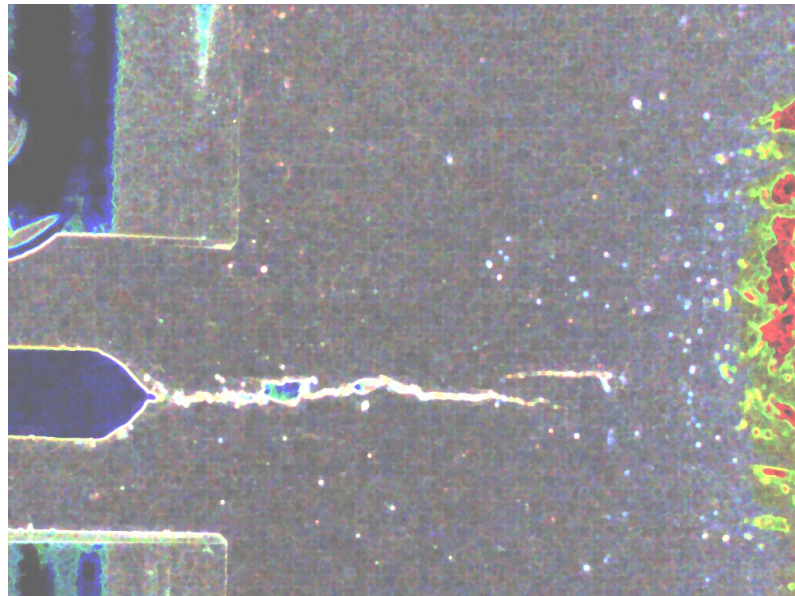


Figure 4.11: Contrast stretching and saturation

Adaptive Gaussian thresholding on the resulting image

After applying functions like erode, dilate, gradient a threshold based image has to be made. Thanks to the fact that the threshold is applied to three channels (instead to one in case of grey-scale image), better effectiveness is reached. The image has still some objects left, but most of the unwanted data is in other color than black. The black objects: crack and dots, can be subtracted from the background and then the dots can be filtered out.

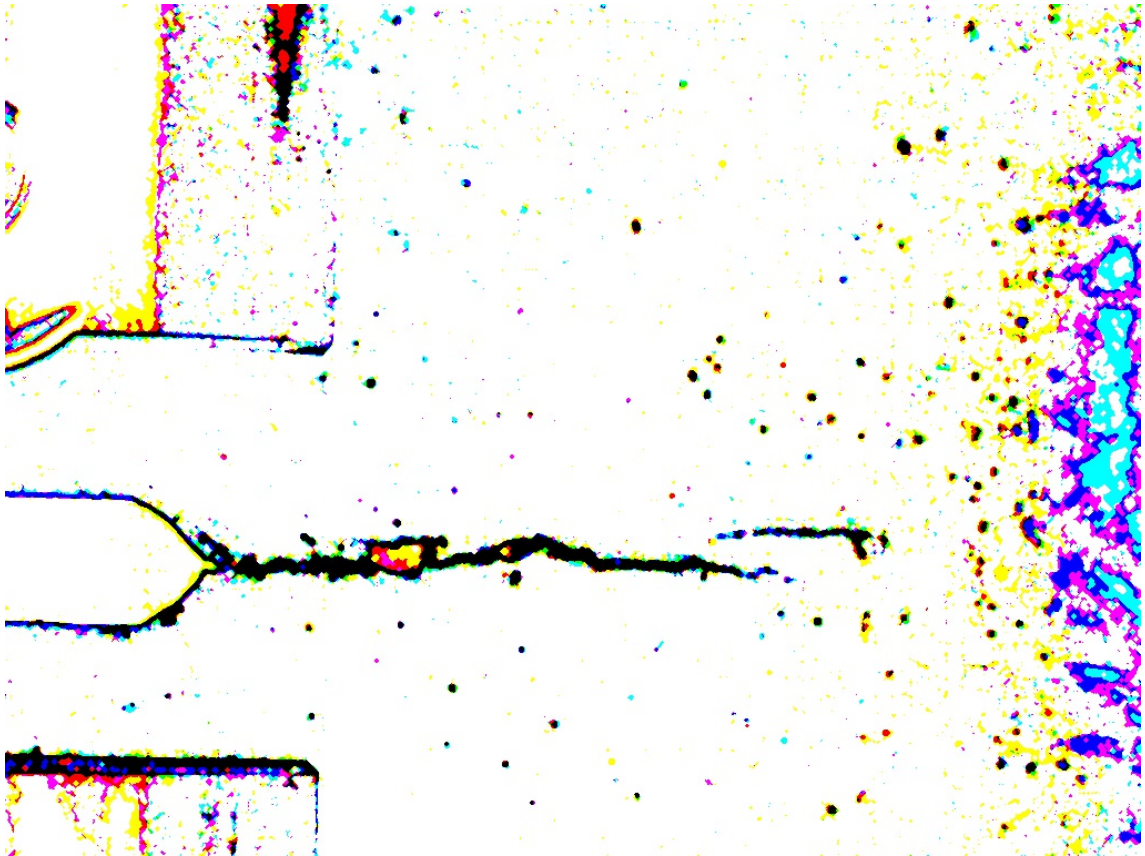


Figure 4.12: Preprocessed image after thresholding operation

This result of the preprocessing is sufficient for the real image processing and although there is still influence of the surrounding, the crack is clearly separated from the background.

4.3 Processing

The processing step is the same for both algorithms. First task is to run the template matching using a template loaded by the user and the original image. This operation obtains the starting point of the crack. This point is saved and passed to the preprocessed image and then this image is segmented into small sub images.

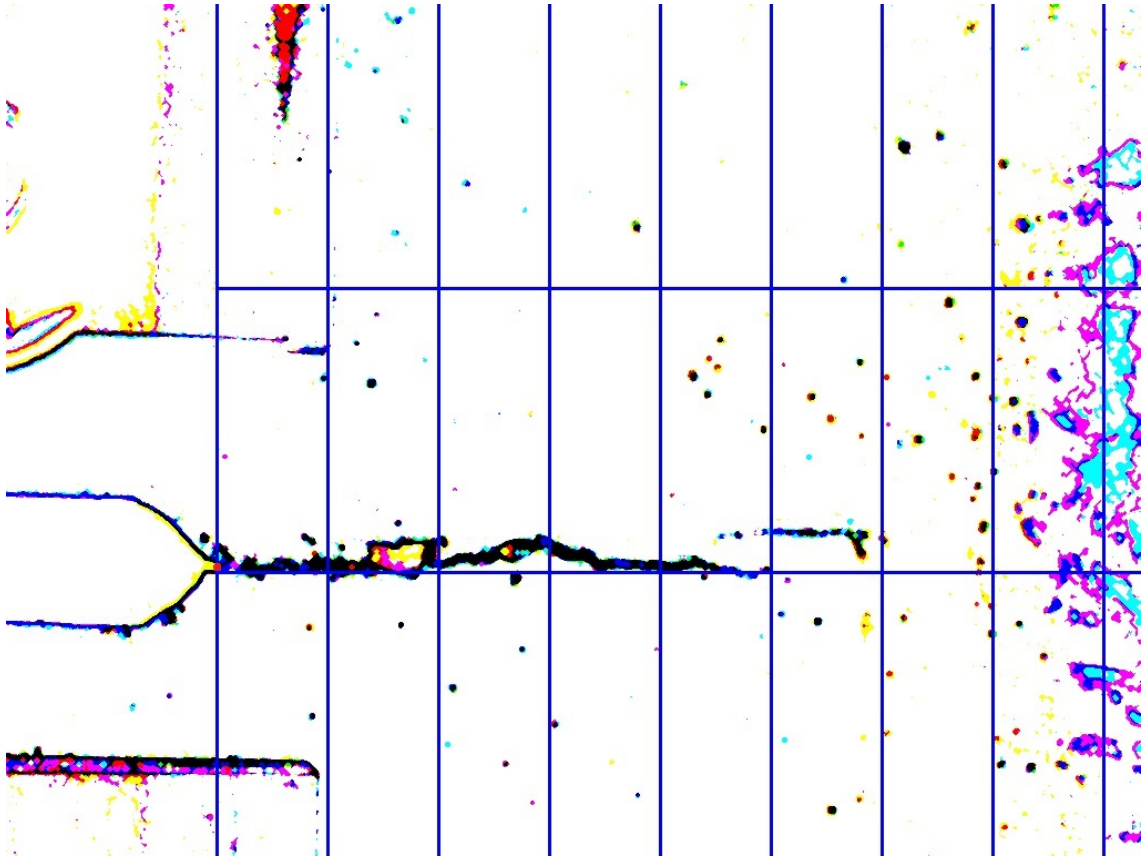


Figure 4.13: Segmentation after template matching

With the information about the start of the crack the algorithm has enough information to make a segmentation of the rest objects and the crack by the use of modified mean filter.

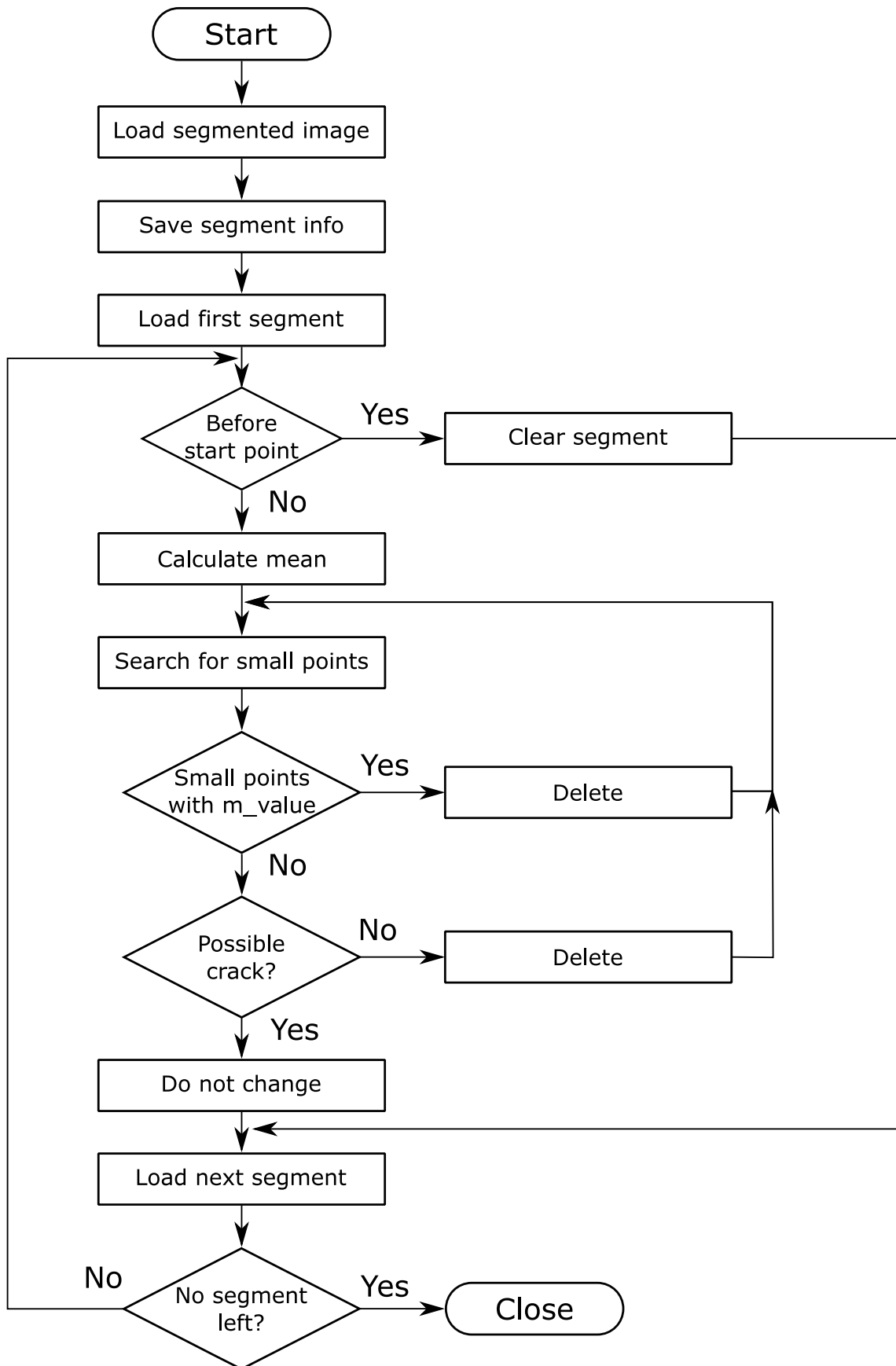


Figure 4.14: Segmentation filter

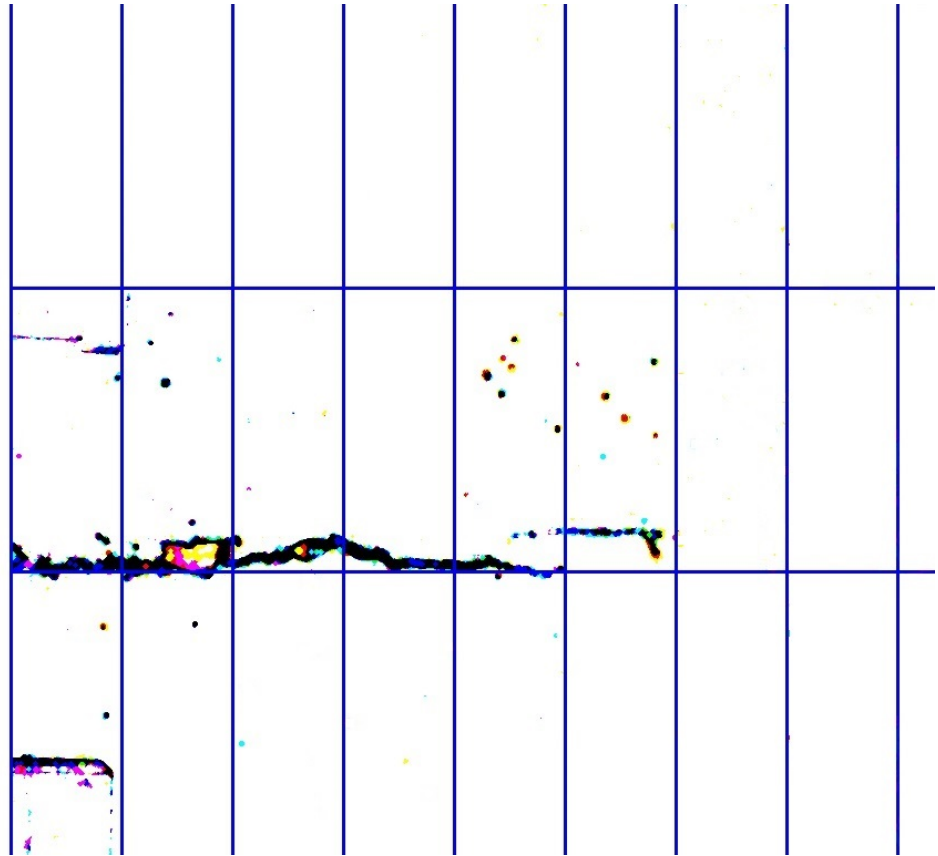


Figure 4.15: Segmented image after the use of modified mean filter

This filter influences only segments with low crack probability. The result is that most of the small objects are eliminated and the crack reconstruction algorithm has less false informations. Last step of the processing is to reconstruct the crack itself. For this method are both preprocessed images used and the result is a propagation path of the crack.

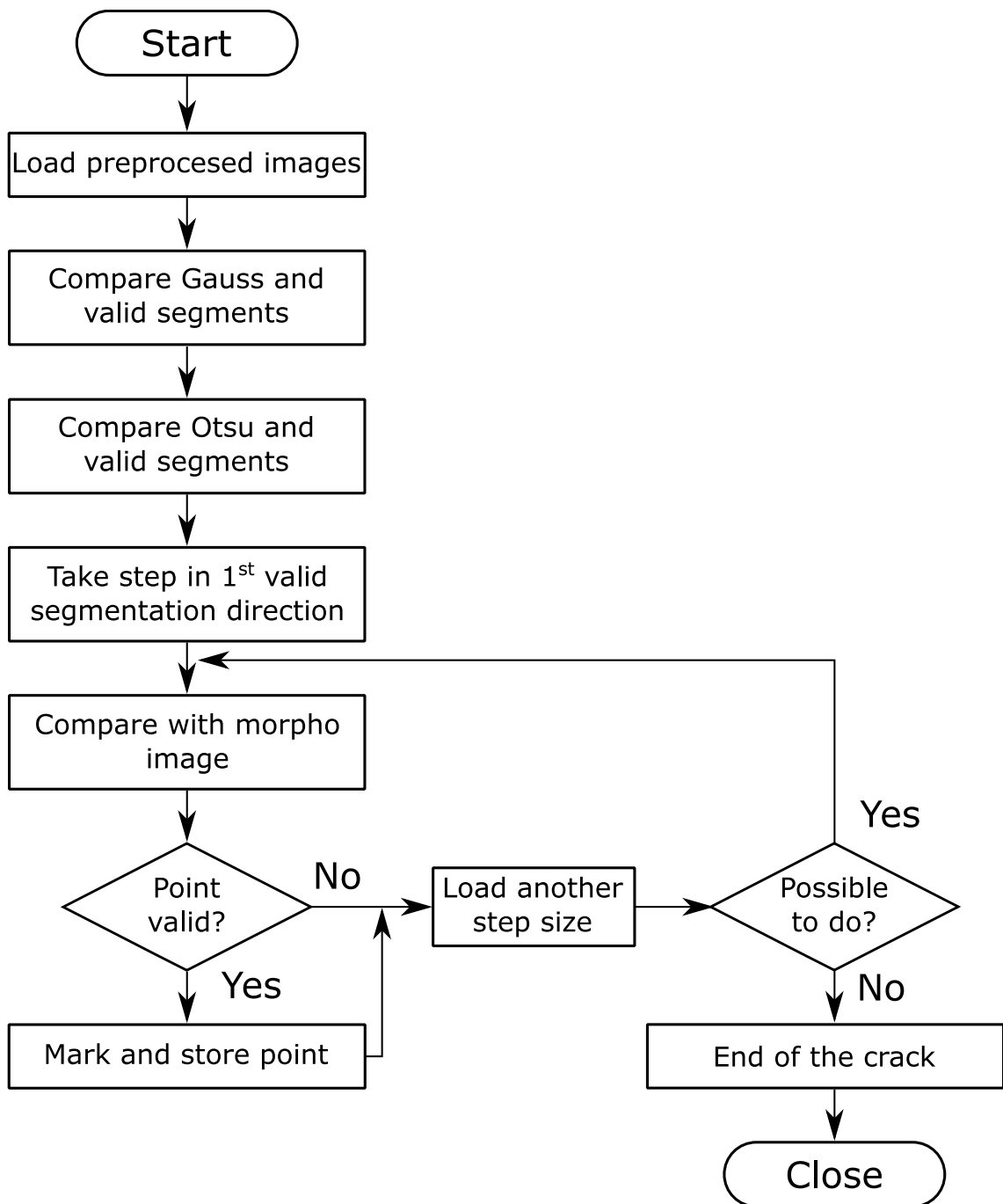


Figure 4.16: Crack reconstruction

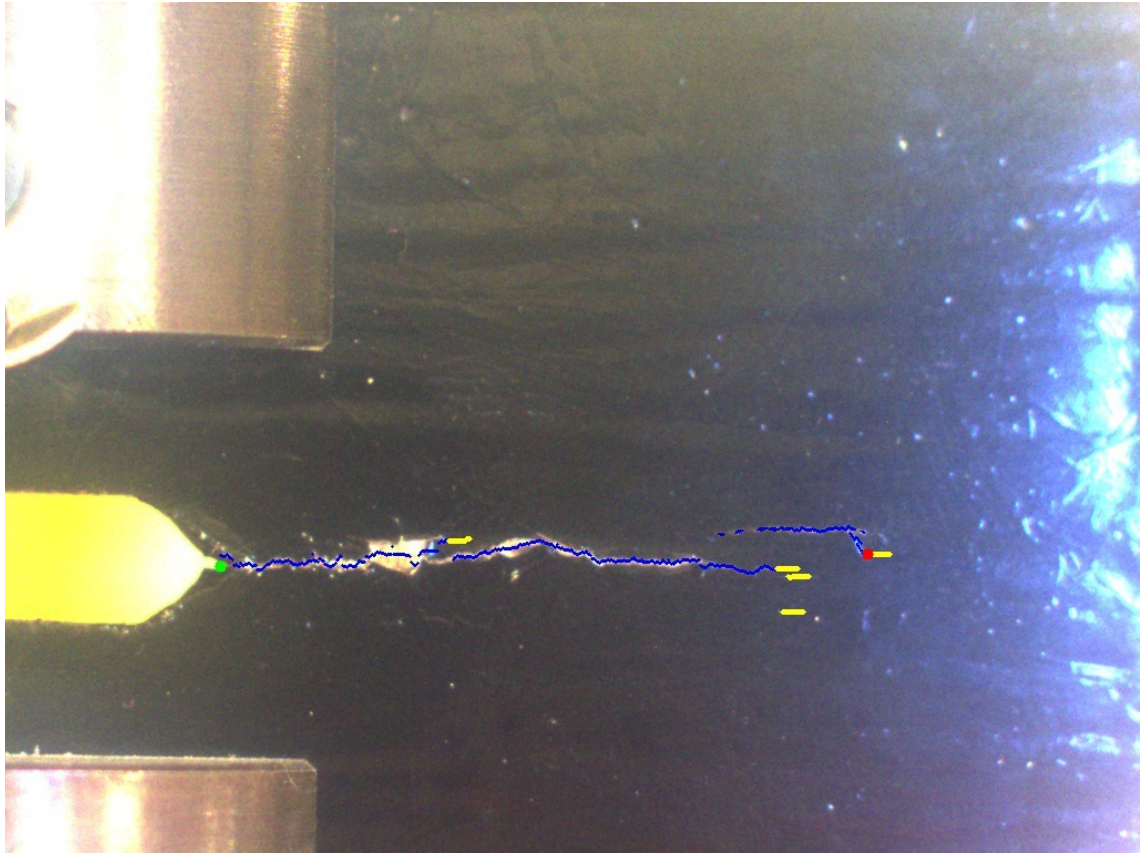


Figure 4.17: Reconstructed path

In the image above it can be seen the propagation of the crack highlighted with the blue colored curve. This crack starts at the green point obtained by template matching and ends with the red point obtained by the reconstruction algorithm. The figure 4.17 shows additionally yellow lines. These lines are blind crack paths or small objects recognized as possible crack part. Without the segmentation and the use of the filter, there would be many more blind paths.

4.4 Post-processing

Main task of post processing is to evaluate the acquired data about the crack. First function of post processing is to evaluate the length and height of the crack. This is done by the help of the data about the scaling obtained by preprocessing and the data about the beginning and end of the crack obtained in processing. This part of the work is divided into several steps:

1. **Draw lines:** To draw lines is the first step of post processing. Straight lines are drawn through points obtained from the reconstructed path. If the values of the path points are relatively in same horizontal range, they are considered a line. When the points range varies, then the line is not vertical, a drawing function is called and the next point is calculated as $y_{point} = a \cdot x_{point} + b$.
2. **Calculate length:** The length of the drawn lines is given in pixels. This value is recalculated using the data obtained in preprocessing to get a number in SI unit. This value slightly differs in vertical and horizontal direction (the mean value with the tested set-up was one millimetre is 7.5 pixels in horizontal direction and 6.5 pixels in vertical direction).
3. **Calculate angles:** After the second step it is possible to calculate the angles between each drawn line and the horizontal direction. For this step elementary mathematical functions contained in standard math library (sine, cosine) are used.
4. **Calculate properties:** To calculate properties of the crack it is necessary to know information about time, initial crack length, initial load cycles of the test probe and tested frequency. The first two parameters can be obtained from the image and the following two parameters must be given by the user (table 4.1).

Table 4.1: Properties

Input parameters

| | |
|---------------------------------|------|
| Initial crack length a_0 [mm] | 12.5 |
| Initial load cycles N_0 [-] | 0 |

Output data

| | |
|---|---|
| Crack length a [mm] | |
| Load cycles N [-] | $N = f \cdot t$ |
| Crack increment da [mm] | $da = a(t_{i+j}) - a(t_i)$ default: $j = 1$ |
| Load cycle difference dN [-] | $dN = N(t_{i+j}) - N(t_i)$ default: $j = 1$ |
| Crack growth rate $\frac{da}{dN}$ [$\frac{mm}{cycles}$] | $\frac{da}{dN} = \frac{a(t_{i+j}) - a(t_i)}{N(t_{i+j}) - N(t_i)}$ |

5. **Print image:** After completing all the steps before, the processed image is shown with basic information written in it. For example:

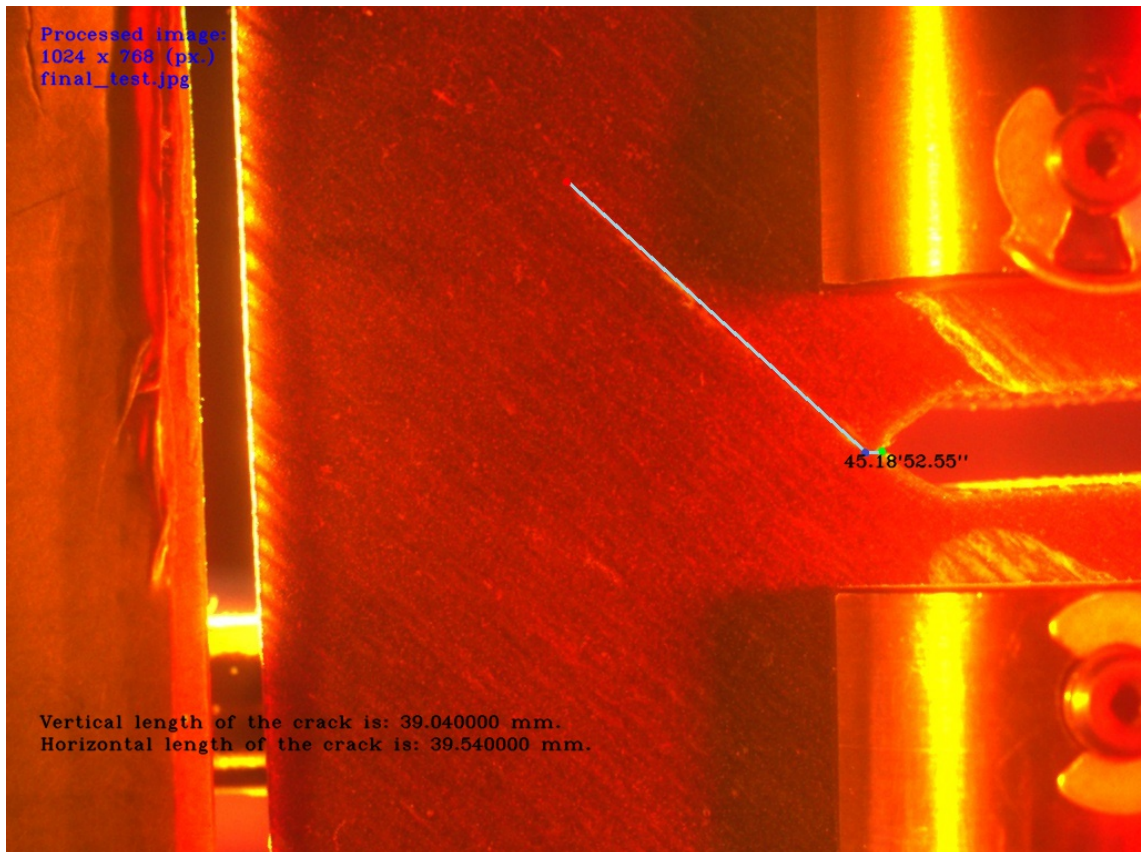


Figure 4.18: Output image

6. **Save image info:** The rest of the information needed for evaluation of the material properties are saved in a file (image.info.dat). This file contains such data:

Table 4.2: Output file

| | | | | |
|--------------------|----------------|--------|------------|-------------------------------------|
| Image name: | image_name | .jpg | | |
| Resolution: | width | height | | |
| Date: | Day\Month\Year | Hours | | |
| a[mm] | N[cycles] | da[mm] | dN[cycles] | $\frac{da}{dN} [\frac{mm}{cycles}]$ |
| • | • | • | • | • |
| • | • | • | • | • |
| • | • | • | • | • |
| • | • | • | • | • |

Printing of results into a file and into the final image is the last step of the post processing.

5 Result presentation

Most of the results are included as an attachment on the DVD. The results and influence of the filters with the previously used camera objectives can be also found on the DVD. The data show that orange and yellow coloured filters have the best performance in this particular case. This is an expected result. As mentioned in section 3.1.3 (Filters), orange and yellow filters are used for contrast enhancement. Results obtained with the use of red coloured filters are, contrary to expectations, useless. Red coloured filters should also enhance contrast but in this case they reduce the contrast of the crack path and leave other objects without a change. Results obtained by the acquisition program are also included on the DVD. The amount of the data makes it impossible to show all results as a part of the printed copy of this thesis.

The test stand was changed in the end therefore they are only four sets of the new testing data. The data can be processed with the presented program, but to achieve a really good results in all cases, there is a need to set the light system, set the aperture and try another filters. The filters used for the last experiments are sufficient but they were chosen randomly (only the color of the filter was chosen by experience). In this thesis are old (figure 4.3) and new (figure 4.1) samples used for the explanation of the developed functions. In the last section 4.4 is the image 4.18 taken from the last experiment without properly set aperture. The algorithm used to process this data had to use all implemented functions such as advanced search. This is a limit case for the correct function of the algorithm without any modifications. In case of the first test where the crack grow in horizontal direction another setup of the test stand was used (aperture, light) and the results were better. In case of the vertical growth there is a problem with shadows (also visible on figure 4.18) from the probe attachment. These shadows are causing that part of the cracks cannot be evaluated (this crack has empty spaces). In other words, it is possible to evaluate the length and the angle of about 60% of the sampled images (at the beginning and at the end it is possible). This loss in probes makes it impossible to evaluate the

whole process of growth. After the test aperture settings were changed to see the influence of the settings and on some of these images it is possible to evaluate the whole crack.

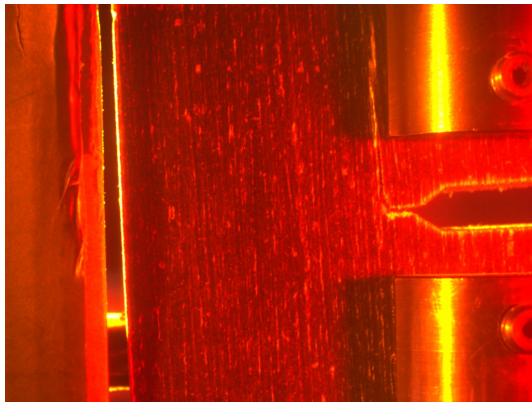


Figure 5.1: Darker image

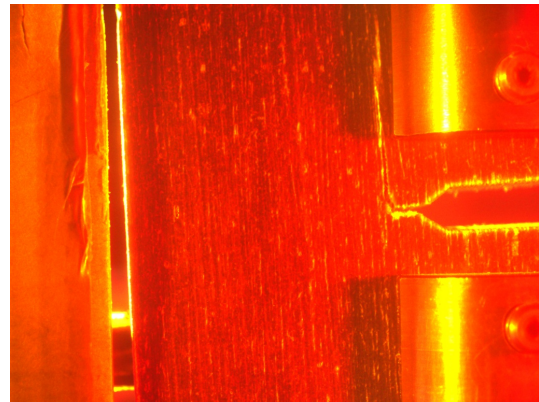


Figure 5.2: Brighter image

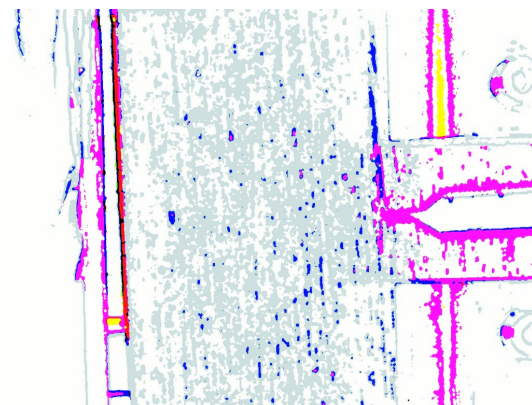


Figure 5.3: Threshold of darker image

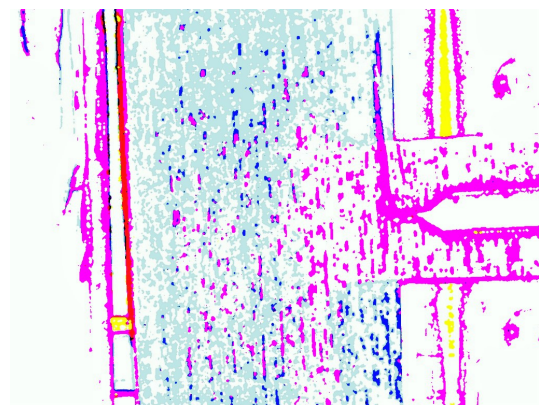


Figure 5.4: Threshold of brighter image

The brighter image is easier to recognize for the human eye, the crack is visible and it seems like it would be a better choice for evaluation. But the reality is different, the darker image has less objects for the segmentation filter and the crack is more isolated from the rest of the background. The difference between these two images is only 0.8 mm in length evaluation (brighter better) but the propagation can be easier observed on the darker image. This makes the evaluation of the crack in the growth stadium more precise. It is possible that the results could be improved by use of difference light filters or with a different position of the light sources.

The capturing program could be further developed as the uEye camera has more than one hundred possible settings which could affect the results of the data analysis.

The two most important features are embedded in this version (v0.81). The first feature is auto gain. This function calculates the mean value of the incoming light and sets a digital aperture (makes the image brighter or darker). This feature is active on all cams on in default mode (in the webcams it is impossible to disable it). Auto gain function devaluates the aperture on the objective and the installed light sources and it is possible to switch it off. The second feature is white balancing. This function tries to keep the same value of white and therefore devaluates the used filters. This function can be turned off.

6 Conclusion

Automatic crack analysis is a powerful tool which can save time for the evaluation of the data from fracture mechanic experiments. In case of this work two programs were developed, one for the acquiring of the test probes images and one for the evaluation of the data. Both programs are part of one installation package as they require a package of libraries and register data for a proper run. Both programs are using only open source and newly created libraries for they function thus it is possible to distribute them under an open source licence.

Image acquisition program is able to use any camera listed under Windows as recording device and any type of uEye camera without the need to install the IDS acquisition software or drivers. This is possible thanks to the implementation of the dynamic and static libraries from the IDS SDK into the installation program. This program has at its disposal basic settings for the camera, which enhance the advantages of the industrial cameras (such as the possibility to set the color depth, auto white balancing and automatic gain control of the CCD for better performance).

Software for image analysis enables processing of the the acquired images and estimation the crack propagation and its properties. Because of the use of OpenCV 3.0.0, this framework would have to be installed on the targeted computer, taking over 3GB of space. For this reasons are the needed libraries compiled and set as a part of the installation program. This step reduces the requirement of disk space to 120MB (just for the libraries). The analysis program produces great results for most cases and selective filter can sort out most of the unwanted data. The program uses separate functions for the processing of the acquired images and it can be further modified. There are two void functions implemented for the crack pathing that can be filled with new algorithm or with an enhancing function for the analysis, without the need to rewrite the program itself. Both created programs have been often modified due to the changes in equipment and new test samples. Each program has its own changelog listed in the appendixes B and C.

A Contents of enclosed DVD

DVD

- |— Master_thesis_Latex latex source files
- |— Master_thesis_PDF pdf output
- |— Crack Capture program
 - |— inc
 - |— src
- |— Images produced Crack Capture program
- |— Crack Analysis program
 - |— inc
 - |— src
- |— Result_presentation

B Changelog of Capture program

Capture program

| Version | Changes | Reason |
|---------|--|--|
| 0.2 | Added uEye camera option | New camera on test stand |
| 0.3 | Added possibility to have two uEye cameras | New camera on test stand |
| 0.4 | Fixed freezing of uEye camera; Added function for reading camera and sensor info; | Added function is...Freeze() as a wait function for image loading from CCD into memory; To have information about the origin of the image; |
| 0.5 | Added GUI | |
| 0.6 | Added function uEye/Webcam and Use | To have a possibility to use one or two cameras and to have the possibility to combine them with other recording devices |
| 0.7 | Fixed opening of webcam window after quitting the uEye recording window | Memory was not automatically deallocated |
| 0.8 | Added auto gain function; Added auto white function; | To have a possibility to turn on/off the gain function (like digital aperture); To have a possibility to turn on/off the white balancing (for better filter results); |
| 0.81 | Fixed saving into a dedicated folder; Check if the folder empty is; | Tested computer didn't have C:\; To not overwrite the old test samples; |

Figure B.1: Changelog: Capture program

C Changelog of CrackAnalyzer program

| Version | Changes | Reason |
|---------|---|--|
| v0.1 | Added algorithm from Max Eisenbeiß | |
| v0.2 | Added algorithm for CrackTracking | |
| v0.3 | Added algorithm for Segmentation | Better filter possibilities |
| v0.4 | Changed algorithm for CrackTracking | Use of colored images |
| v0.5 | Added selective filter | New cameras with better resolution |
| v0.6 | Modified selective filter Modified segmentation method Modified algorithm for CrackTracking Added advanced search function Added probability function Added search for scaling Added method for writing output data | Smaller cracks than the first test probes Smaller cracks than the first test probes Smaller cracks than the first test probes Scaling paper not always on same spot |

Figure C.1: Changelog: Capture program

Bibliography

- [1] R.T.W. Calvard et al. *Picture thresholding using an iterative selection method*. Transactions on Systems Man and Cybernetics, 8(Aug):630–632. Aug. 1978.
- [2] Paul Bourke. *RGB colour space*. Available at: http://paulbourke.net/texture_colour/colourspace/(Accessed 3.7.2016). Mar. 1995.
- [3] Daniel Martín Carabias. *Analysis of image thresholding methods for their application to augmented reality environments*. Master thesis on Universidad Complutense de Madrid, Available at: http://eprints.sim.ucm.es/16932/1/Tesis_Master_Daniel_Martin_Carabias.pdf(Accessed 4.7.2016). June 2012.
- [4] Erik Dahlstrom et al. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. Available at: <https://www.w3.org/TR/SVG/single-page.html>(Accessed 5.7.2016). Aug. 2011.
- [5] Edward R. Dougherty and Roberto A. Lotufo. *Hands-on Morphological Image Processing*. SPIE Publications, July 2013. ISBN: 978-0819447203.
- [6] David Duce. *Portable Network Graphics (PNG) Specification (Second Edition)*. Mar. 2003.
- [7] Daniel Durini. *High Performance Silicon Imaging: Fundamentals and Applications of CMOS and CCD sensors*. first edition. Woodhead publishing, Feb. 2014. ISBN: 978-0857095985.
- [8] Adrian Kaehler Gary Bradski. *Learning OpenCV*. First Edition. O’Reilly Media, Inc., Sept. 2008. ISBN: 978-0-596-51613-0.
- [9] *Graphics Interchange Format*. Version 89a, Available at: <https://www.w3.org/Graphics/GIF/spec-gif89a.txt>(Accessed 6.7.2016). CompuServe Incorporated Columbus Ohio. Mar. 1990.
- [10] Eric Hamilton. *JPEG File Interchange Format*. Version 1.02, Available at: <https://www.w3.org/Graphics/JPEG/jfif3.pdf>(Accessed 6.7.2016). C-Cube Microsystems. Sept. 1992.
- [11] Gerald C. Holst and Terrence S. Lomheim. *CMOS/CCD Sensors and Camera Systems*. second edition. Spie Press Book, Mar. 2007. ISBN: 978-0819486530.
- [12] JN Kapur, P.K. Sahoo, and AKC Wong. *A new method for gray-level picture thresholding using the entropy of the histogram*. Computer vision, graphics, and image processing, 29(3):273–285. Mar. 1985.
- [13] John Miano. *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*. Addison-Wesley Professional, Aug. 1999. ISBN: 978-0201604436.
- [14] Maria Petrou and Costas Petrou. *Image Processing: The Fundamentals, 2nd Edition*. first edition. Willey, Apr. 2010. ISBN: 978-0-470-74586-1.
- [15] Mehmet Sezgin. *Survey over image thresholding techniques and quantitative performance evaluation*. Journal of Electronic Imaging 13(1), 146–165, Available at: <http://pequan.lip6.fr/~bereziat/pima/2012/seuillage/sezgin04.pdf>(Accessed 4.7.2016). Jan. 2014.
- [16] Stuart A. Taylor. *CCD and CMOS Imaging Array Technologies: Technology Review*. Technical Report EPC-1998-106, Available at: <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CCD.pdf>(Accessed 27.6.2016), Rev.3. Xerox Research Centre Europe. 1998.

- [17] *The OpenCV Reference Manual*. Release 3.0.0-dev, Available at: <http://docs.opencv.org/3.0-beta/opencv2refman.pdf>(Accessed 28.6.2016). OpenCV. July 2016.
- [18] *The OpenCV Tutorials*. Release 2.4.13.0, Available at: http://docs.opencv.org/2.4/opencv_tutorials.pdf(Accessed 1.7.2016). OpenCV. July 2016.
- [19] W.H. Tsai. *Moment-preserving thresholding: A new approach*. Computer Vision, Graphics, and Image Processing, 29(3):377–393, Mar. 1985.
- [20] *User Manual uEye Software Development Kit (SDK)*. Available at: http://master-ivi.univ-lille1.fr/fichiers/Cours/uEye_SDK_manual_enu.pdf(Accessed 3.7.2016). IDS Imaging Development Systems GmbH. Sept. 2008.
- [21] *Using GPU Nodes*. Article ID: 298, Available at: http://www.nas.nasa.gov/hecc/support/kb/using-gpu-nodes_298.html(Accessed 1.7.2016). Nasa. June 2016.
- [22] Nicholas Wilt. *The Cuda Handbook: A comprehensive Guide to GPU Programming*. First Edition. Addison Wesley, June 2013. ISBN: 978-0-321-80946-9.