

Vision Soil Analyzer

Product design of a vision based soil analyzer

Jelle Spijker

Copyright © 2015 Jelle Spijker
PUBLISHED BY ROYAL IHC

WWW.IHCMERWEDE.COM
WWW.MTIHOLLAND.COM
WWW.HAN.NL

This document remains the property of “IHC Holland B.V.” All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from “IHC Holland B.V.”

First printing, September 2015



Contents

1	Introduction	7
----------	---------------------	----------

I

Design

2	Functional Design	11
2.1	Global Input-Proces-Output	11
2.2	Specifications	12
2.2.1	Functional requirements	12
2.2.2	Technical requirements	12
3	User Interface	13
3.1	Graphical User Interface	13
3.2	Hardware User Interface	13
4	Manuals	15
4.1	User manual	15
4.2	Administrator manual	15
5	Technical Design	17
5.1	Hierarchical structure	17
5.2	Architecture	17
5.3	Detailed Input-Process-Output schematics	17
5.3.1	Led driver	17
5.3.2	Global position unit	17

5.3.3	Controller	17
6	Vision design	19

II

Realization

7	Technical Realization	23
7.0.4	Electrical design	23
7.0.5	Design	23
8	Vision realization	25
8.1	Image acquisition	25
8.2	Image enhancement	26
8.3	Feature extraction	27
8.3.1	CIE La*b* extraction	28
8.3.2	Fast Fourier Descriptors	28
8.3.3	Particle Size Distribution	28
8.4	Classification	28
8.4.1	Roundness using Hu-moments	28
8.4.2	Angularity using a Neural Network	28
8.4.3	Genetic Algorithm	28

III

Verification

9	Presenting Information	33
9.1	Table	33
9.2	Figure	33

IV

Addenda

Bibliography	37
Books	37
Reports	37
Articles	37
Index	39
A Graphical User Interface	41
B Example Soil Report	45
C RDM campus: Student project	51
D Current project status	53

E	Soil and computer vision flyer	61
F	SoilMath Library	77
F.0.1	Genetic Algorithm Class	77
F.0.2	Fast Fourier Transform Class	88
F.0.3	Neural Network Class	95
F.0.4	Statistical Class	103
F.0.5	General project files	119
G	Hardware Library	131
G.0.6	Microscope Class	131
G.0.7	Beaglebone Black Class	141
G.0.8	GPIO Class	145
G.0.9	PWM Class	152
G.0.10	ADC Class	157
G.0.11	EC12P Class	162
G.0.12	eQep Class	166
G.0.13	SoilCape Class	173
G.0.14	USB Class	174
G.0.15	General project files	176
H	Vision Library	183
H.0.16	Image processing Class	183
H.0.17	Conversion Class	191
H.0.18	Enhance Class	198
H.0.19	Morphological filter Class	206
H.0.20	Segment Class	211
H.0.21	General project files	232
I	Analyzer Library	239
I.0.22	Analyzer Class	239
I.0.23	Sample Class	250
I.0.24	Particle Class	255
I.0.25	Settings Class	261
I.0.26	General project files	266
J	QOpenCVQT Library	269
K	QParticleDisplay Library	273
L	QParticleSelector Library	279
M	QReportGenerator Library	285
N	Vision Soil Analyzer Program	299
N.0.27	General project files	299
N.0.28	Main window Class	303
N.0.29	Dialog window Class	319
N.0.30	Dialog Neural Network Class	334



1. Introduction

This project finds its roots in the minor Embedded Vision Design @ HAN, hereafter named EVD. During this minor an embedded device was developed which analyses soil samples using a microscope. This Vision Soil Analyzer hereafter referred to as VSA, analyzes samples using the optical properties. It gives an user information on color, texture and structure.

This is developed in collaboration with Royal IHC and MTI Holland. Royal IHC is one of Holland major shipyard companies and specializes in dredging and offshore. MTI Holland BV is royal IHC dredging knowledge center. They're worldwide leading centre of expertise in the area of translating knowledge of dredging, mining and deep-sea mining processes into the specification, design and application of equipment.

Both companies have an interests in knowing the properties of soil, be it to advise their customers or to further facilitate their own research and services. Current methods, like the Particle Size Analysis using a sieve and hydrometer are time consuming and non portable. To facilitate quick, accurate and on location soil research an embedded device has been developed. This VSA analyzes soil samples using a microscope and gives the user acceptable and quick results on the soil visual properties.

Quick and reliable results are a welcome addition into any laboratory, this combined with a device that is light and portable gives it's users an added benefit of shortened logistical operations for their soil samples. This results in some serious time benefits.

During the first period of the minor a basic prototype has been developed. This prototype ran in Matlab on a X64 desktop computer and was a first test case for the algorithms and idea's. In the second period this prototype is developed on an ARMv7 embedded Linux device and is compiled in C++. The goal of the software is to analyze soil samples and presenting the user with information regarding it's color, texture and structure.

Information regarding the color of a sample is presented to the user in the CIE Lab and Redness Index color-models. These color models show correlation between different soil properties, such as iron content and fertility. Conversion between different color-models

are CPU intensive, because each pixel will be transformed using multiple algorithms. It's therefore paramount that calculations are done with a minimum of machine instructions and with acceptable errors.

Texture information is presented to a user via a Particle Size Distribution, hereafter named PSD. This is a cumulative function representing the ratio of different particle sizes in the soil sample. Due to the nature of a two dimensional digital image numerous problems arise. These are overlap of smaller particles by bigger particles, this gives a distortion in the PSD results, because the smaller particle is registered as part of the bigger particle. And another problem is the fact that soil particles are three dimensional, but the image is two dimensional.

Information about the structure of the soil is extrapolated from the individual particles shapes. These shapes are described in the frequency domain, using a Fast Fourier Transform and fed into a Neural Network which classifies these shapes into standard soil categories. These are time consuming operations and therefore should be done with a minimum of machine instructions and efficient programming.

This wiki / product documentation gives the developer(s) and customers, namely MTI and IHC a tool to further the development of the VSA into a full fledged market ready product. The development environment and the used protocols are described in order to guard the quality of the work. The product itself is designed by determining a global IPO Input-Process-Output diagram. This leads to the functional specifications. To illustrate the working of the device further the User Interface will be designed which will be supplemented with a short manual. All the above design tools will come together in a detailed IPO. Correct working of the device is guaranteed with various testing protocols. The current working principles follows a set global workflow. The vision related algorithms are described in order to determine the most efficient working order. This results in the complete image processing steps

The following project setup is proposed for the release candidate. Future release will follow the roadmap



Design

2	Functional Design	11
2.1	Global Input-Proces-Output	
2.2	Specifications	
3	User Interface	13
3.1	Graphical User Interface	
3.2	Hardware User Interface	
4	Manuals	15
4.1	User manual	
4.2	Administrator manual	
5	Technical Design	17
5.1	Hierarchical structure	
5.2	Architecture	
5.3	Detailed Input-Process-Output schematics	
6	Vision design	19



2. Functional Design

2.1 Global Input-Proces-Output

System Description

The soil sample is dried and the user makes sure the particle don't bond together. A small portion of the sample is placed on a sample plate. Taking care to separate the individual particles as much as possible. The cover is closed and a microscopic camera is positioned, in an environment where the light conditions are controlled.

The embedded Linux device takes a snapshot which is analyzed using the following computer algorithms: First the individual soil particles are identified in the image, using various algorithms, such as adaptive contrast stretch, Gaussian blurring, OTSU – optimal thresholds separation. The color information is determined with various matrix calculations, translating the RGB pixel value to CIE Lab and Redness Index.

The texture information is determined by counting the number of discrete pixels for each individual article. From this the volume is determined. If the scale of each pixel is known, the volume can be given in SI units.

The structure of an individual particle is determined by getting the edge of the pixels. This is done by creating a mask with a morphological erosion algorithm this mask is subtracted of the original image. The contour is translated to a function using the Dijkstra shortest path algorithm. Where each pixel is described as an imaginary complex number representing the radius towards the center of the particle. The vector holding these values are transformed to the frequency space using the Fast Fourier Transformation. The describing complex numbers gained during this transformation are fed into a feedforward Neural Network, which is optimized using Genetic Algorithms and a previously determined learning data set. The output is presented as probability that a certain particle belongs to a predefined category.

The results are presented to the user via a graphical user interface which are shown when the device is hooked to a monitor carrying a HDMI input. It's also possible to present a report in pdf or a native format which can be downloaded from the device using a LAN network device or optional Wi-Fi or Bluetooth. Basic human interaction can be

performed via an on-board encoder, or optional USB keyboard and/or mouse.

Technical system

Prototype of an intelligent soil microscope

Main function

To analyses a dried soil sample, consisting of particle in the range of $0.02[mm] \leq P \leq 2.0[mm]$ and present a user with information regarding color, texture and structure.

2.2 Specifications

2.2.1 Functional requirements

Name Description

Word Definition

Comment Elaboration

2.2.2 Technical requirements

Name Description

Word Definition

Comment Elaboration

3. User Interface

3.1 Graphical User Interface

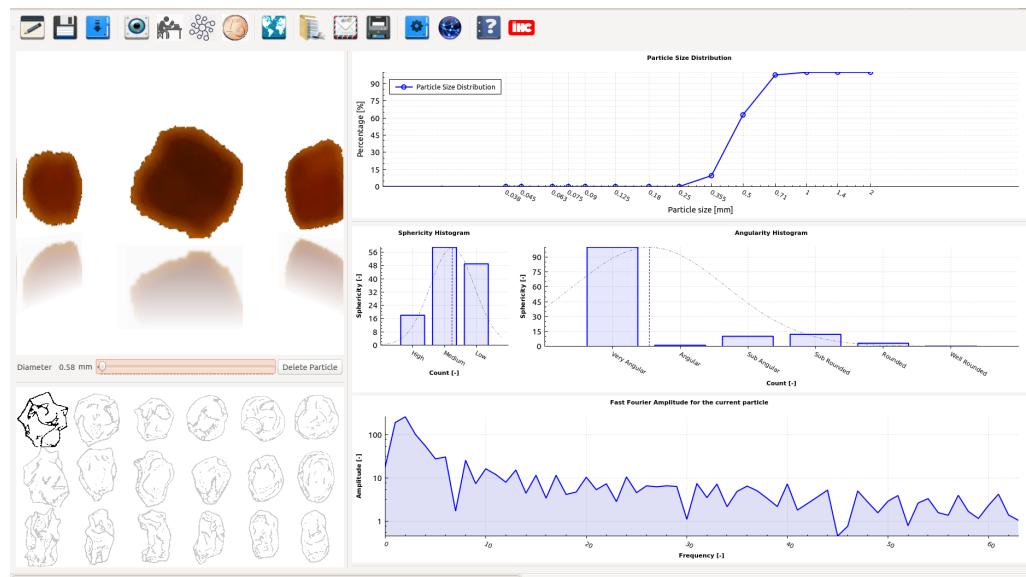


Figure 3.1: Main Graphical User Interface

3.2 Hardware User Interface



4. Manuals

4.1 User manual

4.2 Administrator manual



5. Technical Design

5.1 Hierarchical structure

This is an example of theorems.

5.2 Architecture

This is a theorem consisting of several equations.

5.3 Detailed Input-Process-Output schematics

This is a theorem consisting of just one line.

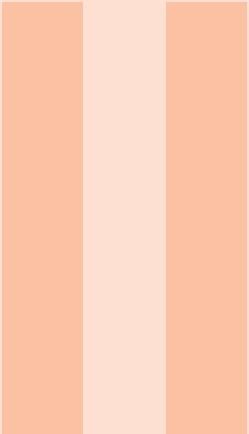
5.3.1 Led driver

5.3.2 Global position unit

5.3.3 Controller



6. Vision design



Realization

7	Technical Realization	23
8	Vision realization	25
8.1	Image acquisition	
8.2	Image enhancement	
8.3	Feature extraction	
8.4	Classification	



7. Technical Realization

7.0.4 Electrical design

7.0.5 Design



8. Vision realization

This chapter describes the used vision processing techniques. The current prototype and work flow is developed to allow for different routines. The user has multiple options and strategies available to achieve optimum results. Each of these are explained in the sequential subsection below. It begins with the acquisition of image(s), which are then enhanced to allow for optimal segmentation of pixels related to sand particles. These pixels are used to determine the features of each particle, which serve as input for the classification algorithms.

8.1 Image acquisition

A thorough review of the current literature [1] identified three properties that can be used in vision based analyzing. These properties are structure (shape), color and texture (size). When looking closely at sand sample, you notice a multitude of shapes, colors and sizes, each particle is unique and differs from its neighbor. This diversity brings it own challenges. The shape of a particle determines how it will rest on the sample plate. The color and the translucency of the particle, determines how easily it can be segmented or identified from the background. Whilst the size determines the needed focus depth of the microscope.

R In samples, where the particles show a huge spread in size, compared to the mean size, there will be a noticeable difference in focus, between big and small particles.

Acquisition strategies

The first prototype is developed in such a way that multiple acquisition strategies can be implemented. Each of these tackle different challenges. The quality of the acquired image is the biggest factor in the successful extraction of a particle, but in order to make any valid claim about the sample, a certain amount of particles have to be examined. To determine the minimum sample size, the following equation can be derived:

Let the reliability be 95% $\therefore z = 1.96$, the probability be $P = 50\%$ and the accuracy be $\alpha = 5\%$; consider the function:

$$z\sqrt{\frac{p \times (1 - P)}{n}} \leq \alpha \rightarrow n \geq \frac{-p \times (P - 1) \times z^2}{\alpha^2} \quad (8.1)$$

This brings the minimum amount of particles to 384. With the predefined range of particle sizes ($0.2[\text{mm}] \leq \text{Size} \leq 2[\text{mm}]$ where P defines a particle) and the limited work area under the microscope, multiple shots have to be taken. Where the sample is rearranged. Between fifteen and twenty shots are usually enough.



The process of rearranging the particles, will be automated in the future. Student of the minor Offshore & Construction taught at the University of Applied Sciences Rotterdam will work on this challenge. This is done on the RDM Campus. This minor starts in September 2015. Their product will serve as input for the second prototype. Their assignment is described in appendix C and will be executed under the auspice of MTI Holland and the author.

Acquisition

Each sample is placed in a light condition room, and laid out on a semitransparent white acrylate plate. The sample can be illuminated with a bright field light source, where the light is aimed directly at an object or the particle can be lit with back lighting. See the course notes [3] for a more in-depth description. The choice for back lighting can be made because translucent particle are harder to segment in a bright field light. The trade off is extra processing time.

After the sample is placed in the light condition room, the microscope takes a image with bright field illumination and, if the option is selected, another one with back lighting. Hereafter the sample is rearranged, this is a manual procedure. Once the sample is rearranged a new set of shots is taken. Each image that is acquired from the microscope is defined by a matrix were the values are triples for the RGB (red, green and blue) values and these are defined by an unsigned byte.

Each image is stored in a vector using a custom container. This container consists of a bright field image, back light image and a SI-conversion factor. Each time the height is changed, the microscope has to be calibrated so that the relation between pixel and [mm] can be determined. This is done by taking a shot of a disc with known dimensions. A single euro cent can serve for this purpose.



The image is stored in the OpenCV matrix (cv::Mat) container. This container is designed to handle image processing data and routines. It makes use of memory management and smart pointers to handle the data effectively.

8.2 Image enhancement

Image enhancement prepares the RGB image for conversion to a binary image. It eliminates noise and brings out wanted features, by using filters.

Intensity image

The first step in this process step is the conversion from the RGB color space to an scalar valued image which represent the luminosity, also known as a intensity image. This luminosity is calculated using a weighted average and is done for bright field and back lit images.

Let \mathbf{I} and $\mathbf{R}, \mathbf{G}, \mathbf{B}$ be a matrices with dimensions $n \times m$ derived from the color matrix \mathbf{RGB} with dimensions $n \times m \times 3$; The weighted average can be calculated with the following equation:

$$\mathbf{I} = 0.2126 \times \mathbf{R} + 0.7152 \times \mathbf{G} + 0.0722 \times \mathbf{B} \quad (8.2)$$

Adaptive contrast stretch

After the conversion from RGB to an intensity image, the user has the choice to apply an adaptive contrast stretch to the bright field images. This process is used to enhance the contrast of the intensity image. For every pixel and its surrounding area the mean and standard deviation are calculated. If the value of the pixel is above or below the mean than the following rule is used to determine the new value: $\mathbf{I}_{n,m} = \mathbf{I}_{n,m} \times \alpha \pm \sigma$, where α is a scaling factor and σ is the standard deviation of the old pixel value with it's neighboring kernel pixels.

Blur

As a second enhancement the user can apply a blurring operation to the bright field images, in essence the opposite of the contrast stretch. The blur operation also determines the mean for every pixels within a given area: the kernel. The mean value of the kernel is assigned to the pixel.

Cropping

The above operations described in the paragraph 8.2 and 8.2, leave the border pixels unaffected in their calculations. This offset is determined by half of the biggest kernel size. These pixels are discarded for the next step. The enhanced intensity matrix is used for particle segmentation, see section 8.3. Whilst the intensity matrix of the bright field image is used for the conversion to the CIE La*b* colorspace, as explained in section 8.3.1.

8.3 Feature extraction

The individual particles have to be identified and segmented from the background. These operations are performed on the enhanced intensity matrix. If the user opted to use back lit and bright field matrices, the enhanced intensity matrices where calculated from the back lit intensity matrices. Otherwise the bright field intensity matrices are used.

Segmentation

The images are segmented by calculating a threshold value. This value is determined by using the Otsu threshold. Xu et al. [2] describe that the Otsu threshold is equal to the average of the mean levels of two classes partitioned by this threshold. This threshold value can be iteratively determined.

Let \vec{h} be a vector of dimension 256 which represent a count of values in the enhanced intensity matrix \mathbf{I} with dimensions $m \times n$

$$\frac{1}{t_o} \sum_{i=1}^{t_o} \vec{h}_i = t_o - \frac{1}{256 - t_o} \sum_{i=t_o}^{256} \vec{h}_i \quad (8.3)$$

In order to get more control over the segmentation process, the normal Otsu's method, as shown above is altered. A user now has the option to choose whether bright or dark object are segmented and how much the intensity values may deviation from the

mean value. This mean value is either the left or right hand side of the equation 8.3 modified with a scaling factor and the standard deviation, as shown in equation 8.4 and 8.5.

Let t_o be the threshold value obtained with the iteration algorithm used to solve equation 8.3, α be the a multiplication factor given by the user and let \vec{h} be a vector of dimension 256 which represent a count of values in the enhanced intensity matrix \mathbf{I} with dimensions $m \times n$

If dark objects are to be obtained

$$t = \frac{1}{t_o} \mu + \frac{1}{2} \alpha \sigma \text{ where } \sigma = \sqrt{\frac{1}{t_o} \sum_{i=1}^t (\vec{h}_i - \mu)^2}, \text{ and } \mu = \frac{1}{t_o} \sum_{i=1}^t \vec{h}_i \quad (8.4)$$

else

$$t = \frac{1}{t_o} \mu - \frac{1}{2} \alpha \sigma \text{ where } \sigma = \sqrt{\frac{1}{256 - t_o} \sum_{i=t}^{256} (\vec{h}_i - \mu)^2}, \text{ and } \mu = \frac{1}{256 - t_o} \sum_{i=t}^{256} \vec{h}_i \quad (8.5)$$

Binary Image

The segmented pix

Let $\mathbf{B} \subset \mathbb{Z}^n \rightarrow \{0, 1\}$

$$0 = 1 \quad (8.6)$$

8.3.1 CIE La*b* extraction

8.3.2 Fast Fourier Descriptors

8.3.3 Particle Size Distribution

The normal procedure for creating a Particle Size Distribution uses sieves and weights, to determine the volume of the the particle.

The Sieve mesh size can be perceived as a cross section of a particle, since the particle is only retained in a sieve when it passes through the top sieve but can't pass through the sieve below. The cross section of the particle is at minimum the sieve mesh size of the top sieve, but other dimensions of the particle can exceed the sieve mesh size at which it last passes.

[Work this argument to explain why it's oke to use a 2 dimensional representation of a 3 dimensional particle]

8.4 Classification

8.4.1 Roundness using Hu-moments

8.4.2 Angularity using a Neural Network

Angularity of particle can be described as

8.4.3 Genetic Algorithm

Viva la revolution!

This is an example of examples.

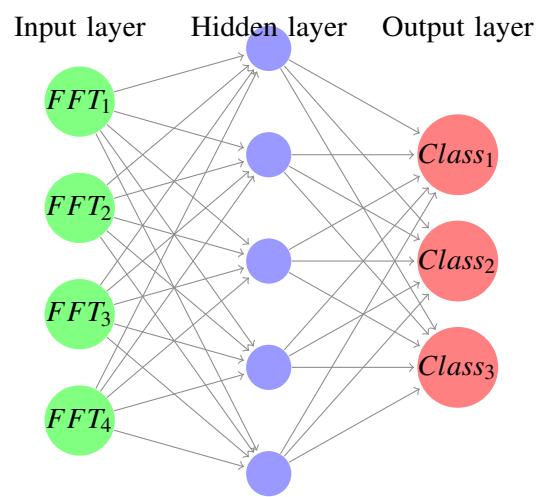
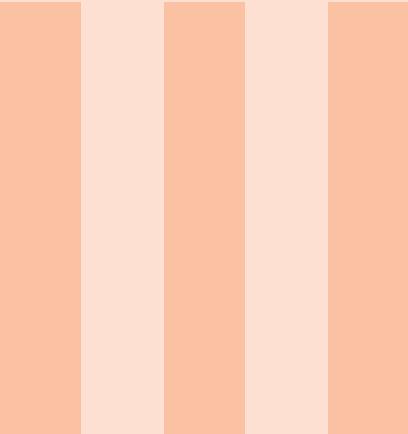


Figure 8.1: Neural Network



Verification



9. Presenting Information

9.1 Table

Treatments	Response 1	Response 2
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table 9.1: Table caption

9.2 Figure

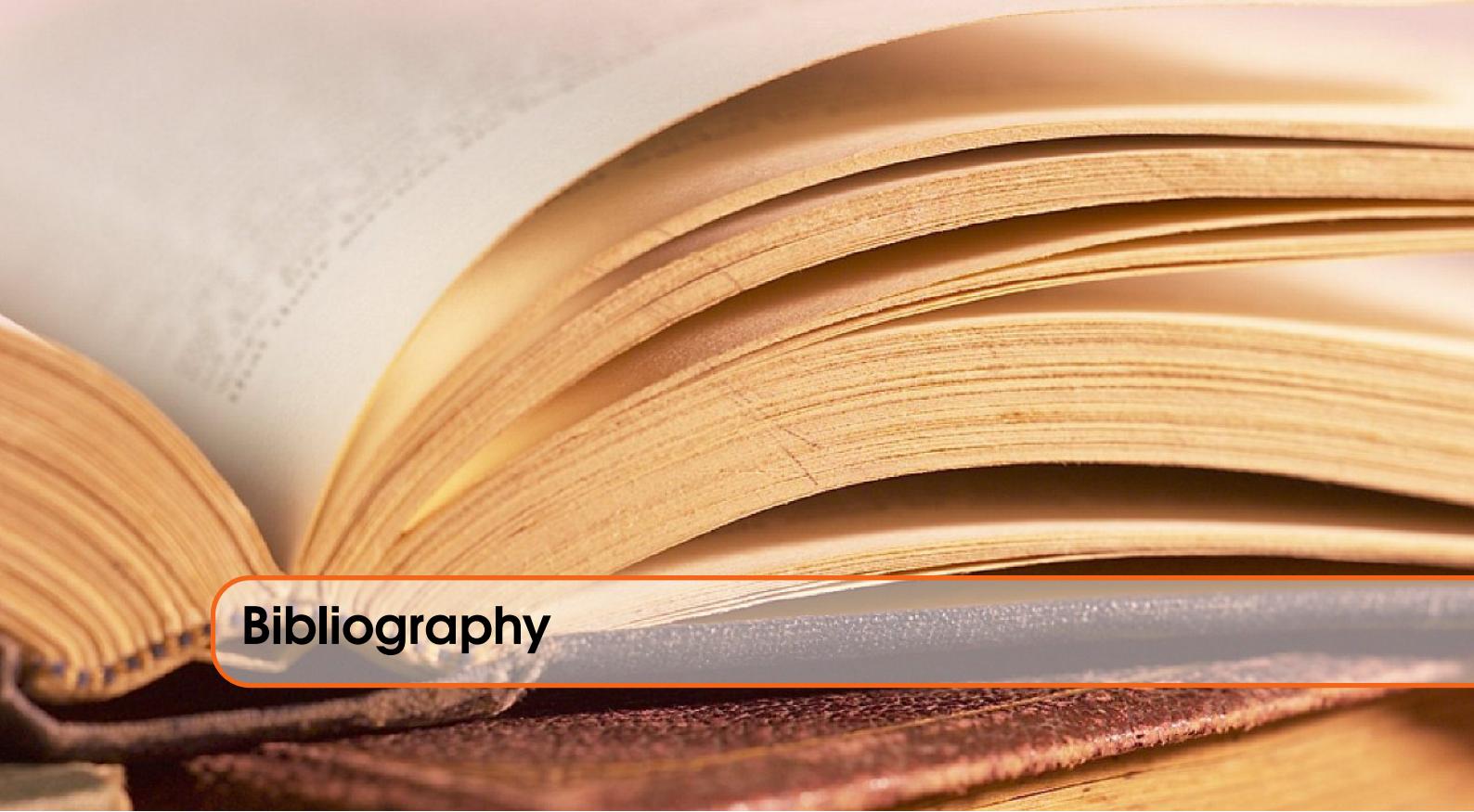


Figure 9.1: Figure caption

V

Addenda

Bibliography	37
Books	
Reports	
Articles	
Index	39
A Graphical User Interface	41
B Example Soil Report	45
C RDM campus: Student project	51
D Current project status	53
E Soil and computer vision flyer	61
F SoilMath Library	77
G Hardware Library	131
H Vision Library	183
I Analyzer Library	239
J QOpenCVQT Library	269
K QParticleDisplay Library	273
L QParticleSelector Library	279
M QReportGenerator Library	285
N Vision Soil Analyzer Program	299



Bibliography

Books

- [3] ir. P.A.C. Ypma. *Course Notes EVD2*. University of applied sciences, Sept. 2, 2014.
71 pages (cited on page 26).

Reports

- [1] Jelle Spijker. *Optische kenmerken van grond gebruikt bij computer vision*. Literatuurstudie. HAN University of applied sciences, 2014 (cited on page 25).

Articles

- [2] Xiangyang Xu et al. “Characteristic analysis of Otsu threshold and its applications”. In: *Pattern Recognition Letters* 32.7 (2011), pages 956 –961. ISSN: 0167-8655.
DOI: <http://dx.doi.org/10.1016/j.patrec.2011.01.021>. URL: <http://www.sciencedirect.com/science/article/pii/S0167865511000365>
(cited on page 27).

SEATTLE
ARTISTSArchitects
r & Associates
110:32/8,114-120 Aug 1

Architects
SEATTLE
Kirk, Paul Hayden, 1914-
Time 69:76 Jan 28, 1957
Blakeley psychiatric group, Womb with
a View
APR 12 1957

Index

- Acquisition, 25, 26
- acquisition strategies, 25
- Angularity
 - Neural Network, 27
- Architecture, 17
- Back lighting, 26
- Bright field illumination, 26
- Citation, 12
- Classification, 27
- Enhancement, 26
- Feature extraction, 27
- Figure, 31
- Global Input-Proces-Output, 11
- Hierachical structure, 17
- Intensity image, 26
- IPO, 17
- Minimum sample size, 25
- RGB, 26
- Roundness
 - Hu-moments, 27
- SI-conversion factor, 26
- structure, 25
- Table, 31
- Texture, 25

A. Graphical User Interface

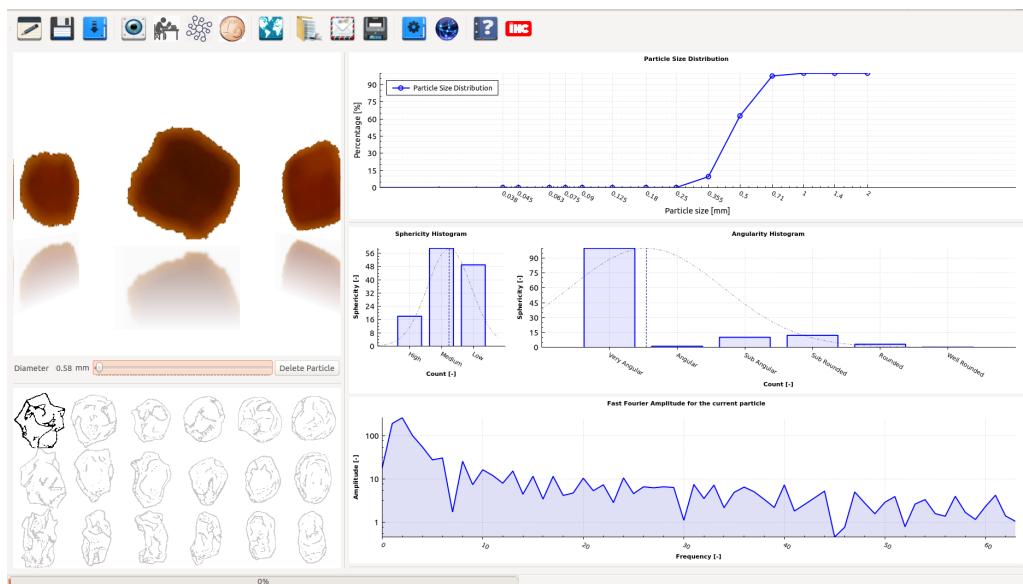


Figure A.1: Main Graphical User Interface

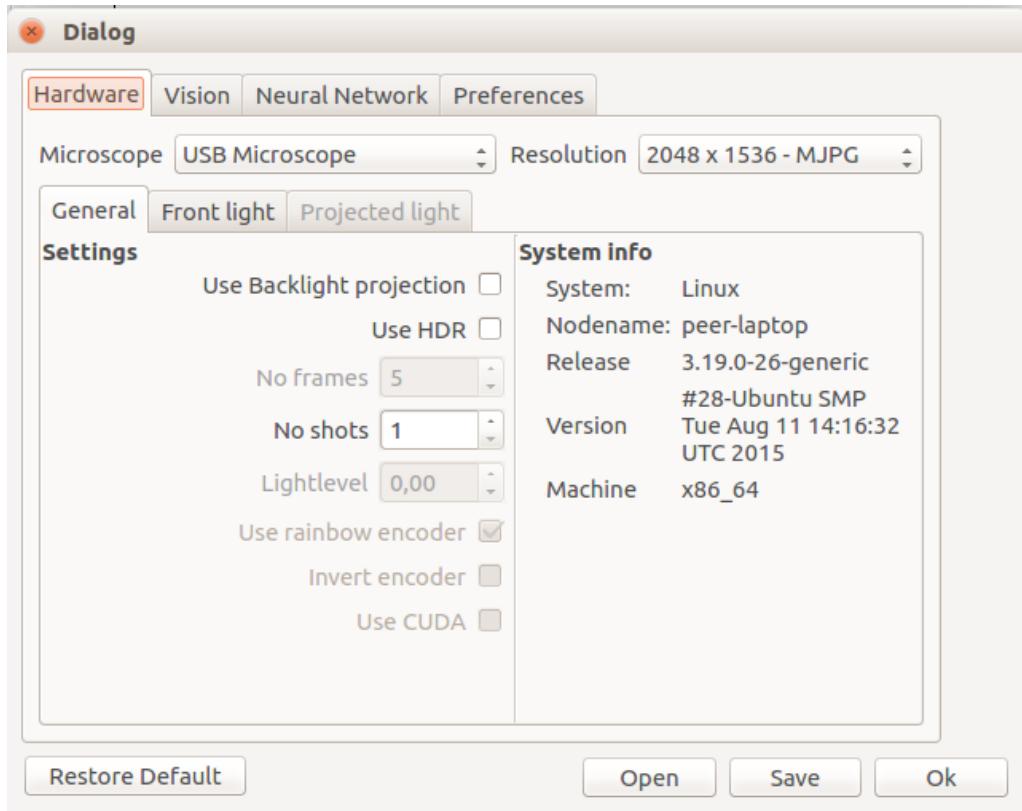


Figure A.2: Settings Hardware Interface

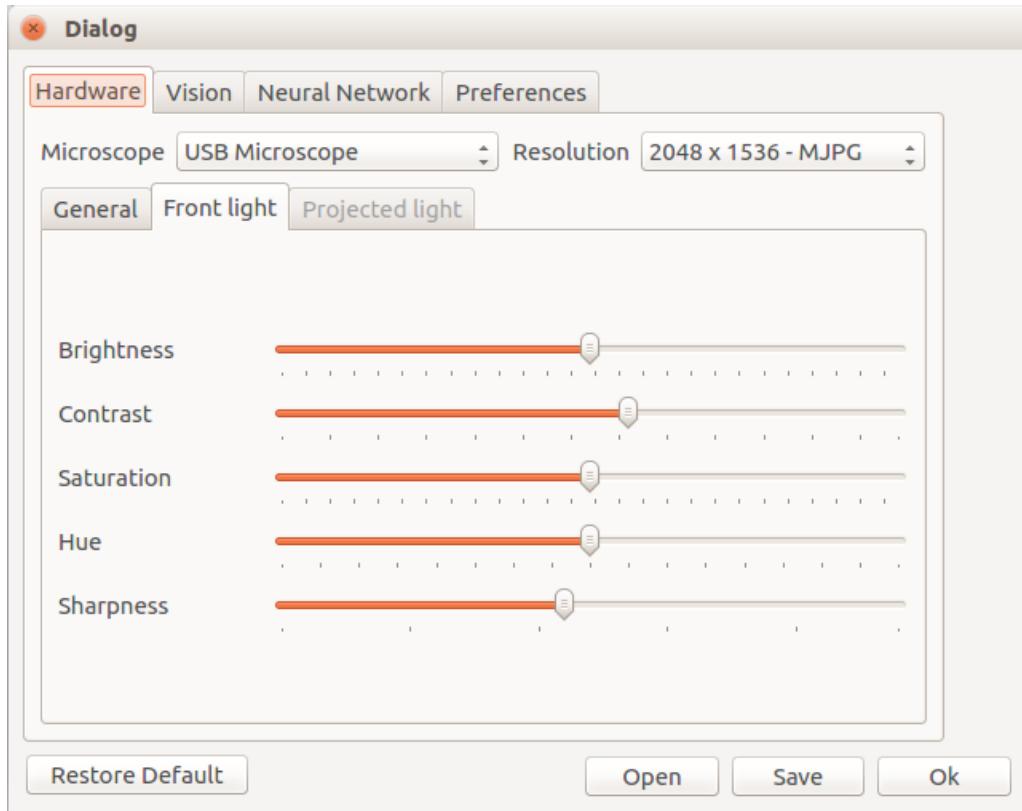


Figure A.3: Settings Hardware Cam Interface

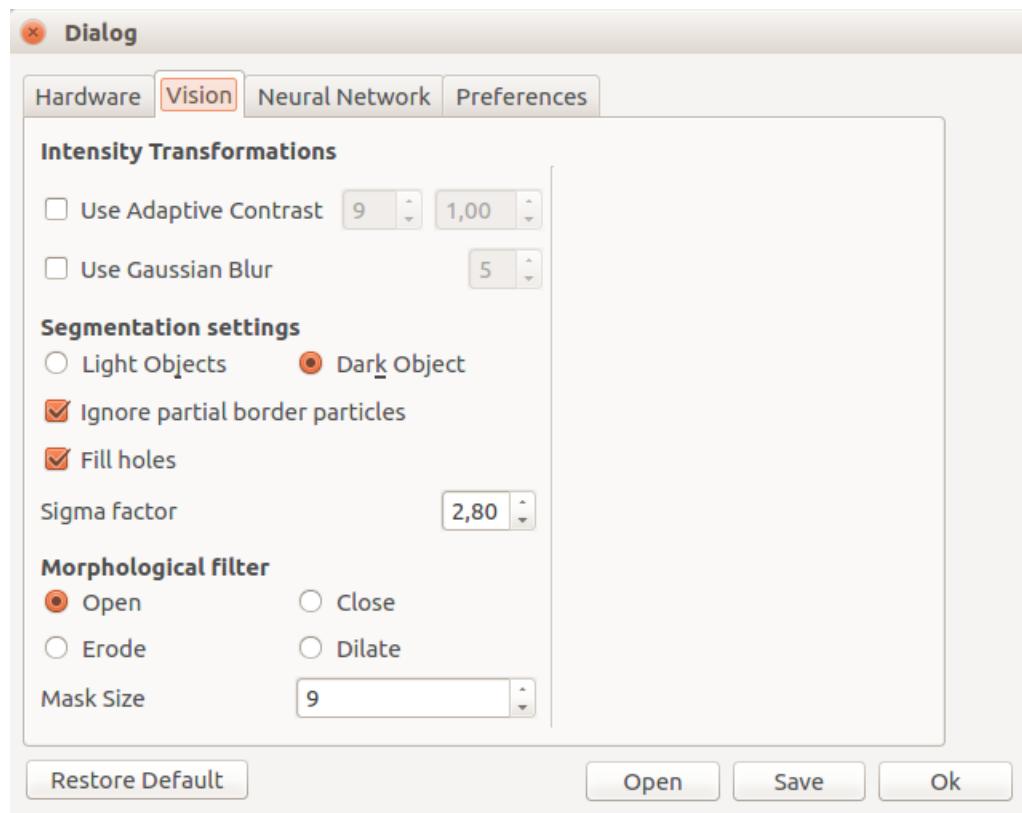


Figure A.4: Settings Vision Interface

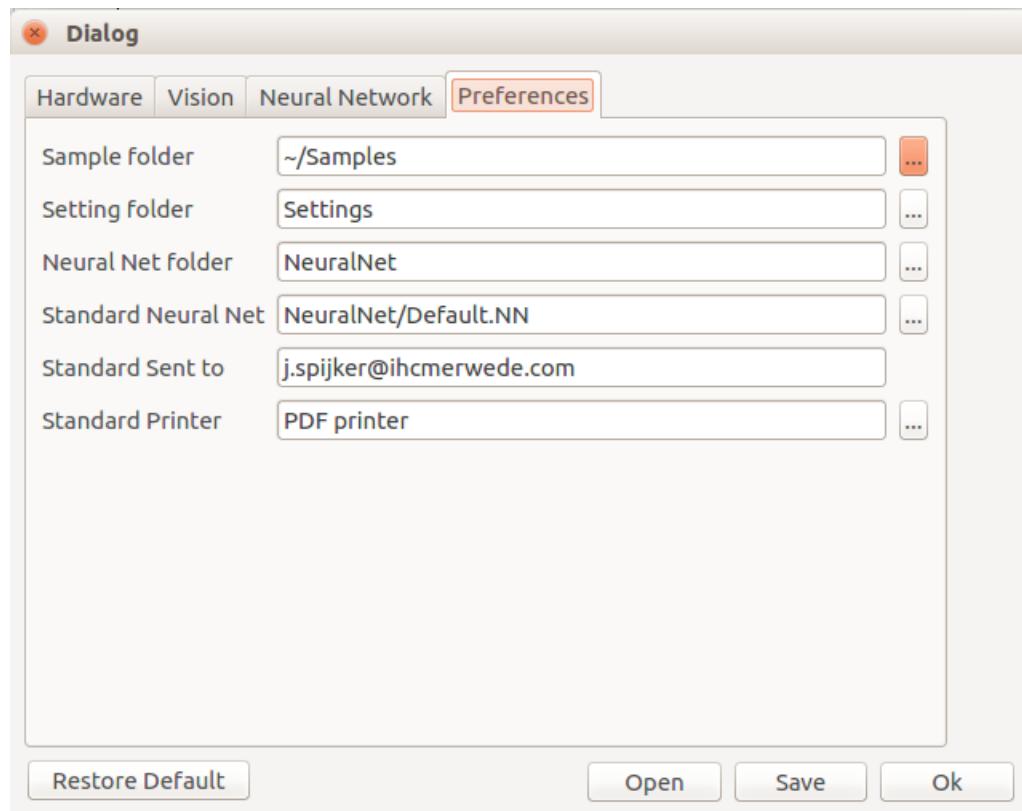


Figure A.5: Settings Preference Interface

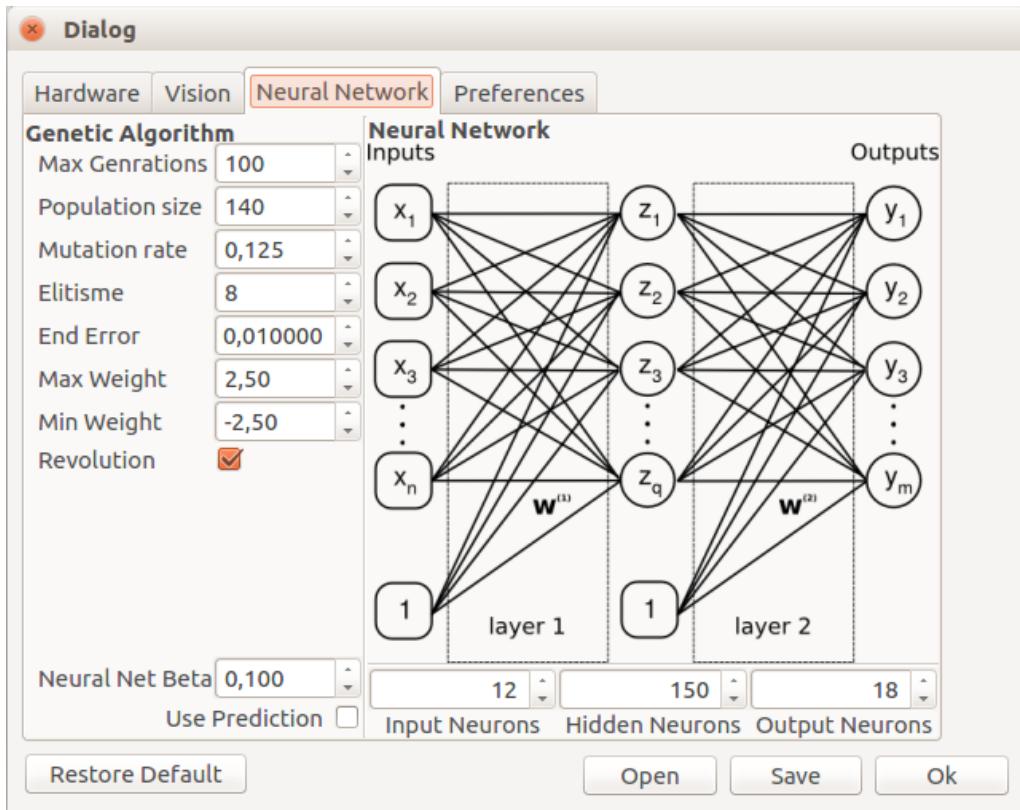


Figure A.6: Settings Neural Network Interface

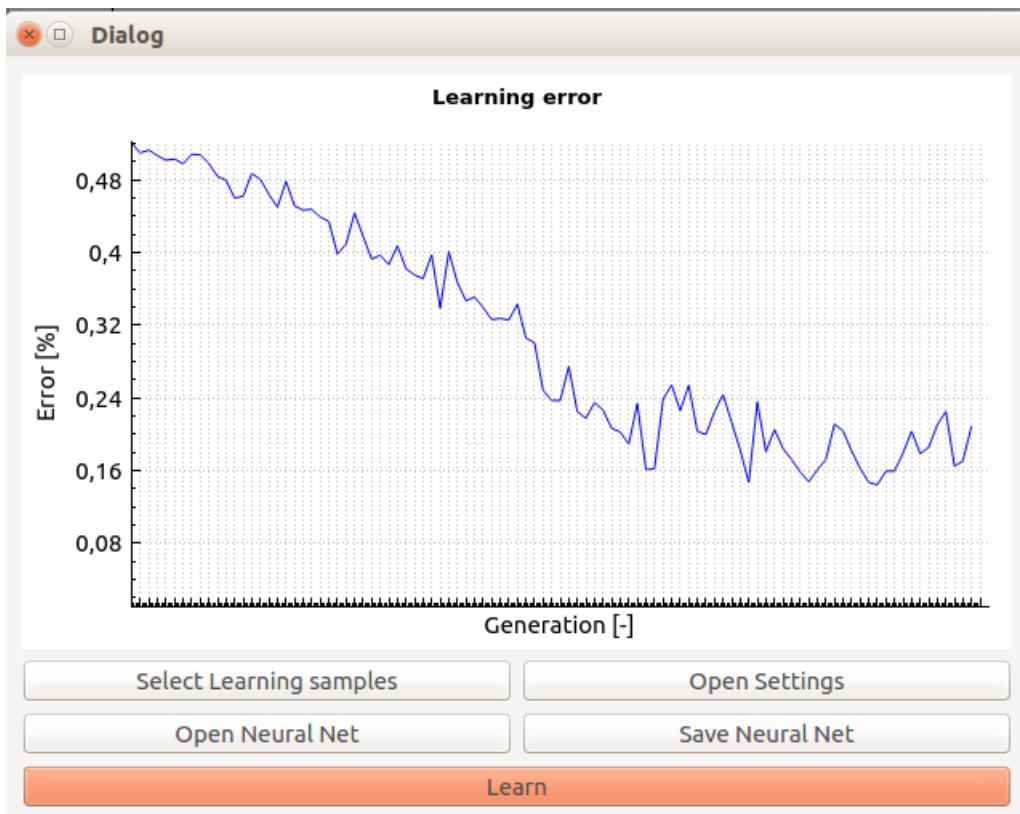
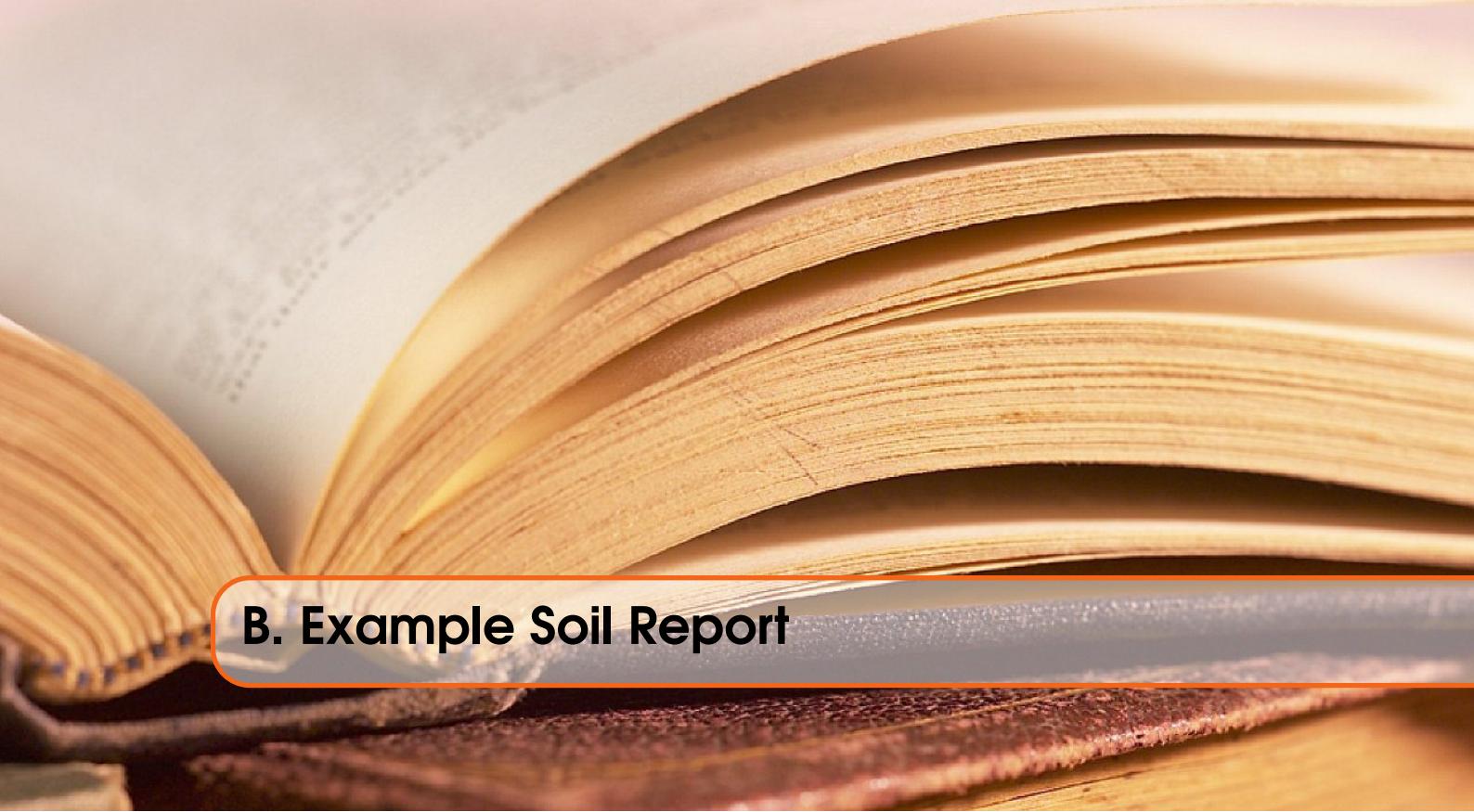


Figure A.7: Neural Network Learning Interface



B. Example Soil Report

Soil Report

Sample name:

Sample ID:

4084628568

Date:

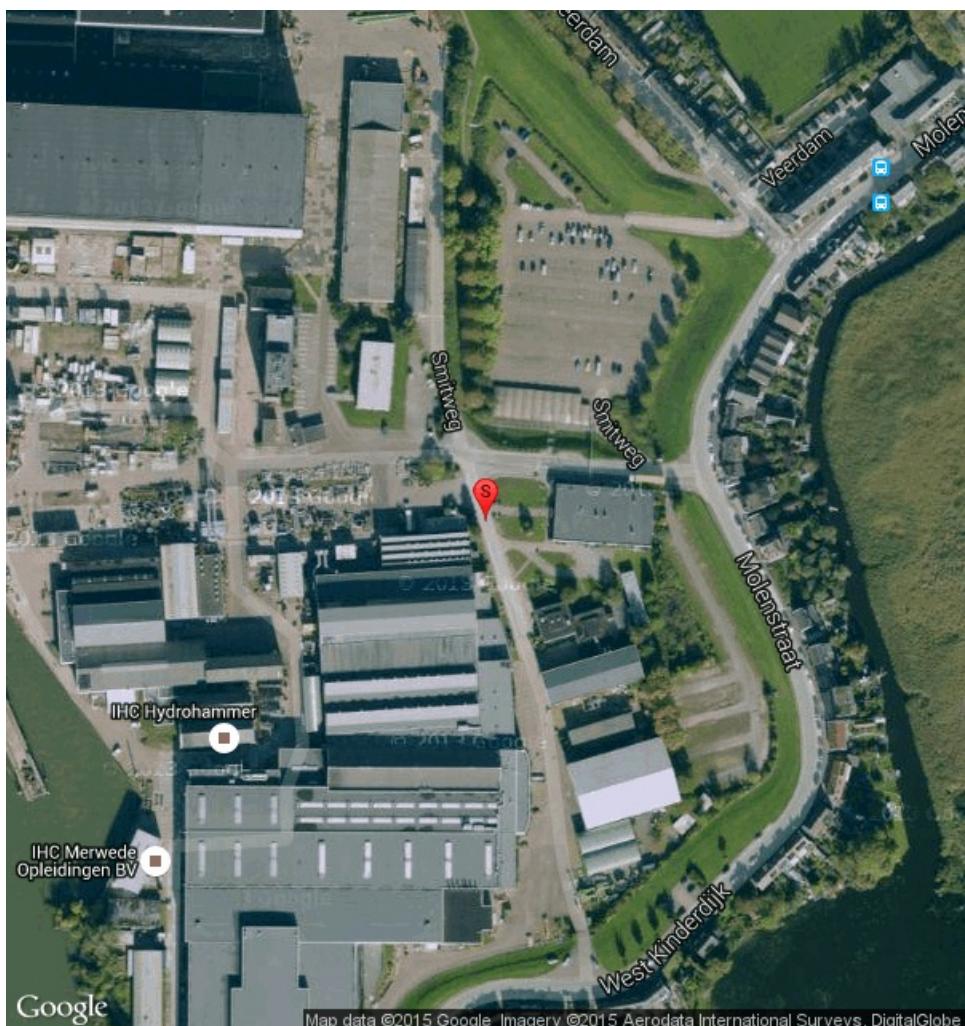
01-09-2015

Location:

51.8849, 4.62962

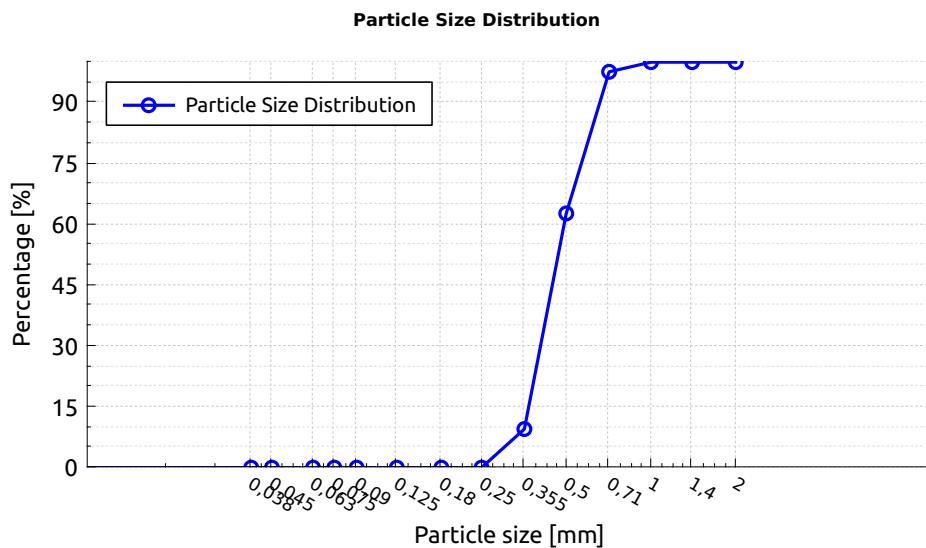
Sample depth:

0 [m]



Particle Size Distribution

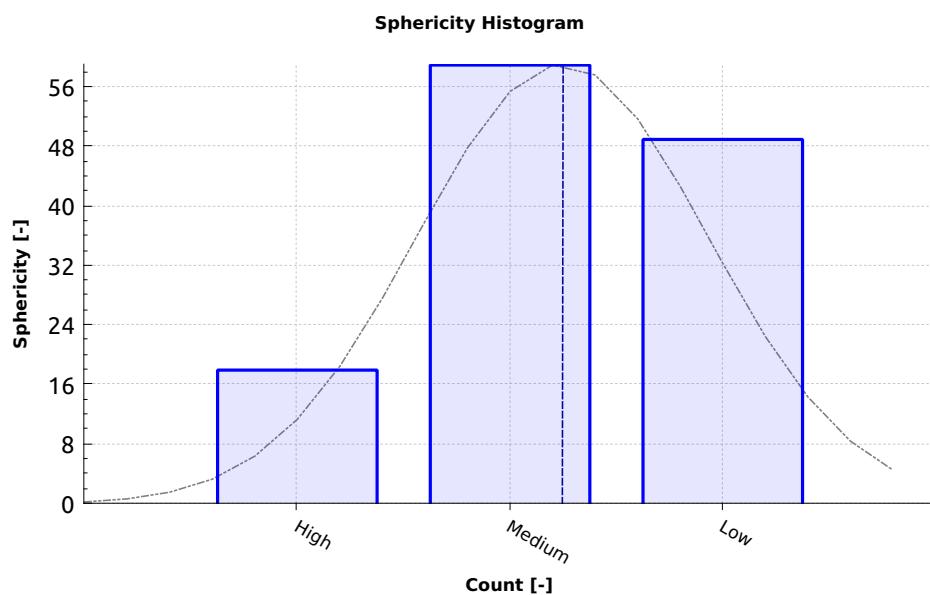
No of particles:	126
Mean:	0.673049
Minimum:	0.407876
Maximum:	1.14742
Range:	0.739541
Standard deviation:	0.142802



Mesh Size [mm]	Cummulatief [%]	Retained [-]
2	100	0
1.4	100	0
1	100	3
0.71	97.619	44
0.5	62.6984	67
0.355	9.52381	12
0.25	0	0
0.18	0	0
0.125	0	0
0.09	0	0
0.075	0	0
0.063	0	0
0.045	0	0
0.038	0	0
0	0	0

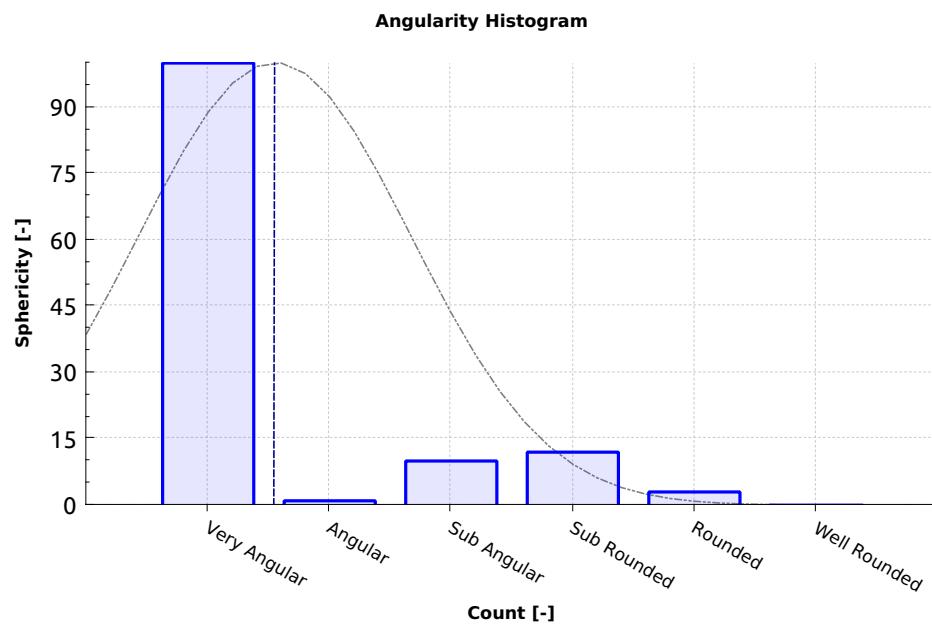
Sphericity Classification

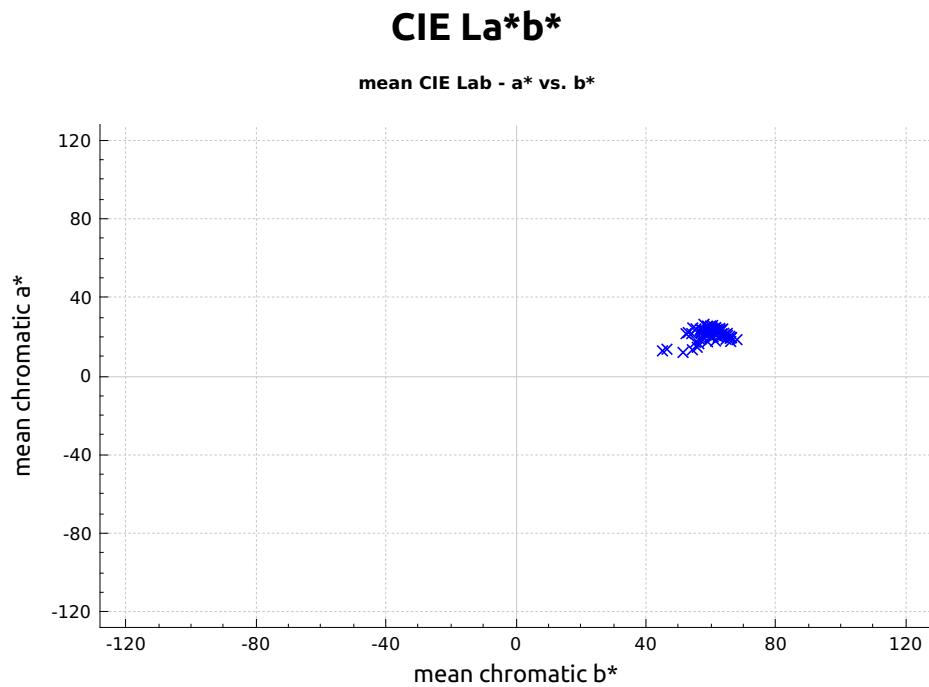
No of particles:	126
Mean:	2.24603
Minimum:	1
Maximum:	3
Range:	2
Standard deviation:	0.686451



Angularity Classification

No of particles:	126
Mean:	1.54762
Minimum:	1
Maximum:	5
Range:	4
Standard deviation:	1.1241







C. RDM campus: Student project

Subject: RDM Student project
Author: Jelle Spijker

Introduction

This project finds its roots in the minor Embedded Vision Design (EVD) taught at the university of applied sciences HAN. During this minor a portable embedded device was developed which analyses soil samples using a microscope. This Vision Soil Analyser hereafter referred to as VSA, analyses soil samples using the optical properties. Its main function is: Presenting quantifiable information to a user on the properties of soil: such as colour, texture and structure.

The VSA takes a snapshot from a soil sample, which is placed under a microscope in an closed environment. This digital image is analysed using a multitude of computer vision algorithms. Statistical data is presented to the user in the form a Particle Size Distribution (PSD) and a histogram of the shape classification. The PSD is obtained by calculating the number of pixels for each individual particle, whilst shape classification is determined by describing the contour of each individual particle as mathematical function which undergoes a transformation to the frequency domain. This complex vector then serves as input for an Artificial Neural Network (ANN) where the output classifies each particle in a certain category.

The prototype developed during the minor EVD will serve as a basis for a graduation project of that same student, which initialized the project. This is done for his main course mechanical engineering at the HAN. This graduation project is done under the auspices of MTI Holland. The goal during this second stage is to develop a field ready prototype. In conjunction with the necessary documentation (Technical Dossier). Due to the scale of the project, several key problems are identified and separated from the main project. These problems can be tackled by separated student groups.

Problem description

Due to the transformation from 3D particles to a discrete 2D image certain data is lost. This degradation of data introduces errors in the statistical data. One of the forms of degradations is the overlap of bigger particle onto smaller particles. These particles are identified as a particle with at least the size and the contour of the biggest particles. Thus giving false negatives for the smaller particles and often false positives for the bigger particle.

A solution that will be explored during this stage is the execution of multiple analysis of the same discrete particle population. This will result in an accurate statistical representation of the soil sample placed under the microscope.

The project that the RDM students can tackle can be described as follow:

Design and build a prototype with which the placement of particles, relative to each other and ranging in sizes from 0.02 - 2 [mm] are randomly changed in a time span of 1 [sec], which is tightly integrated with the main prototype.

The prototype is to be CE compliant and should be build according to technical specifications. It should be described in a Technical Dossier, containing all necessary documents such as: technical drawings (according to mono system), bill of materials, calculation, analysis and design reports.



D. Current project status

Date
15 June 2015

Reference
xxxxxxx

Version
Rev. A

Status
Concept / Final

Name Author
Jelle Spijker

Vision Soil Analyzer

Current status and results



Contact gegevens:

Jelle Spijker (495653) – 06-43272644 – j.spijker@ihcmerwede.com

Disclaimer HAN:

Door ondertekening van dit voorblad, bevestigen wij dat het – door ons ingeleverd(e) werkstuk/rapport/scriptie (verder te noemen "product") – zelfstandig en zonder enige externe hulp door ons is vervaardigd en dat wij op de hoogte zijn van de regels omtrent onregelmatigheden/fraude zoals die vermeld staan in het opleidingsstatuut.

In delen van het product, die letterlijk of bijna letterlijk zijn geciteerd uit externe bronnen (zoals internet, boeken, vakbladen enz.) is dit door ons via een verwijzing conform APA-norm (b.v. voetnoot) explicet kenbaar gemaakt in het geciteerde tekstdeel (cursief gedrukt).

Copyright Royal IHC:

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."



Datum
5 juni 2015

Reference
xxxxxxxx

Version
Rev. A

Pagina
2 van 7

Vision Soil Analyzer

This project finds its roots in the minor Embedded Vision Design taught at the university of applied sciences HAN. During this minor a portable embedded device is being developed which analyses soil samples using a microscope. This Vision Soil Analyzer hereafter referred to as VSA, analyzes soil samples using the optical properties. Its main function is: **Presenting quantifiable information to a user on the properties of soil: such as color, texture and structure.**

Current methods, like the Particle Size Analysis using a sieve and hydrometer are time consuming and non-portable. To facilitate quick, accurate and on location soil research an embedded device has been developed. This VSA analyzes soil samples using a microscope and gives the user acceptable and quick results on the soil visual properties.

Quick and reliable results are a welcome addition into any laboratory, this combined with a device that is light and portable gives its users an added benefit of shortened logistical operations for their soil samples. This results in some serious time benefits.

1.1 Goal

The goal is to develop a device which analyzes soil samples using a digital microscopic camera connected to a microcontroller. The properties that are deemed possible to analyze using this technique are, color, texture and structure. The goal is to perform the calculation within a time span of a five minutes. The results are presented to the user using a generic HDMI monitor or can be download from the device in PDF format. These results fall in to a predefined and for a user acceptable error margin.

1.2 Presented information

The user gets information presented in the following formats:

- | | |
|--------------|---|
| Color | <ul style="list-style-type: none"> • CIE La*b* color model presented as scatterplot with the mean values of each individual particle set out against the chromatic a* and b* axis. Studies indicate a correlation between organic carbon and the values in CIE La*b* color model • Redness Index is presented as statistical data for each individual particle, such as mean, min, max, range, standard deviation etc. Welch tests anova can be executed in order to determine which particle deviates from the rest. |
|--------------|---|

- | | |
|----------------|---|
| Texture | <ul style="list-style-type: none"> • Particle Size Distribution Presented as a cumulative function. These properties show a correlation on water infiltration, pH buffering, buffering of organic materials and much more. |
|----------------|---|

- | | |
|------------------|---|
| Structure | <ul style="list-style-type: none"> • Shape classification regarding each individual particle presented as histogram. The roundness and the angularity are determined and presented as sixteen individual classes. Ranging from high sphericity / well rounded to low sphericity / very angular. These properties show a correlation between erosion, biochemical and physical properties including tool degradation. |
|------------------|---|



Datum
5 juni 2015

Reference
xxxxxxxxx

Version
Rev. A

Pagina
3 van 7

2 First test results

The test setup was a X64 desktop computer running Matlab 2014a. The microscope was placed in an open environment. This setup served as a testing ground for the various algorithms. The goal was to develop a test setup with which to test the various computer algorithms and validate the theory.

Success	Challenges
Segmentation and identification of individual particle is possible for non-transparent particles. See figure 2	Segmentation of transparent particles is a challenge. This is critical to overcome, since systematic exclusion of a certain subset of particles gives inaccurate statistical results
Color model transformation can be strategically performed. Saving calculation time	Variations in color related results. Due to changing light conditions during the day. Test result relating to color could not be reproduced during the day.
Volume of the particles, could be roughly calculated, from a 2D image	Overlap of smaller particles by bigger particles. The combination of the two samples overlapping particles where registered as one bigger particle. This gives distorted PSD results.
Individual particles where identified, and the Fourier Descriptors could be calculated. The inverse of these descriptors translated to accurate results. See figure 1	No correlation between estimated shape classification and human shape classification. The Neural Network was fed an in perfect an small learning dataset.
	Calculation of image consisting of 5e6 pixels takes roughly 7 minutes

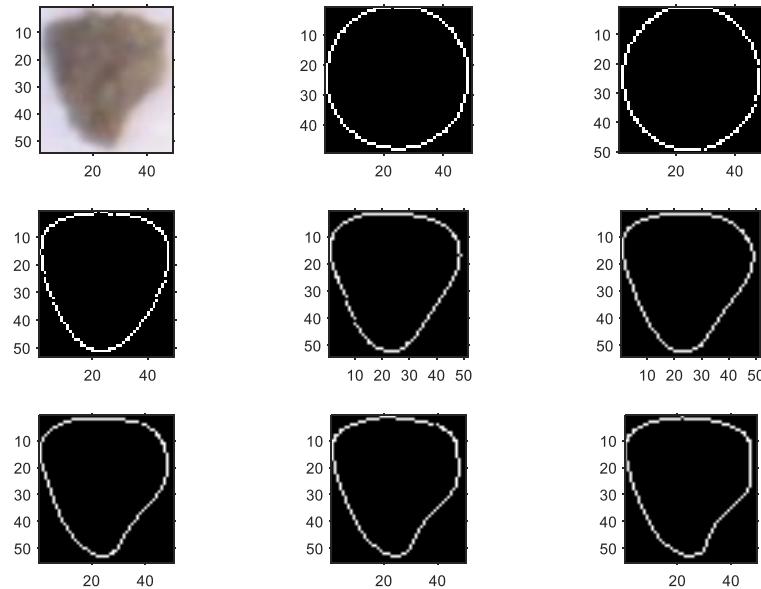


Figure 1 Fourier Descriptors of a soil particle

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

The technology innovator.



Datum
5 juni 2015

Reference
xxxxxxxx

Version
Rev. A

Pagina
4 van 7



Figure 2 Soil particles separated from the background

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

The technology innovator.



Datum
5 juni 2015

Reference
xxxxxxxx

Version
Rev. A

Pagina
5 van 7

3 Second (current) stage

The second stage of this project consist of a transfer from Matlab to C++ running on an embedded Linux ARMv7 device. This consist of rewriting and designing all the algorithms from scratch. Implementing and unit testing them. Design and construction of a light condition case. Design and construction of a PCB for control of the light conditioning case and user interaction. See figure 3 and 4.

The source code currently consists of 7000+ lines. Although the code can be run from a Linux Desktop computer and can probably be ported to a Windows computer. The code and algorithms are designed and optimized for the ARM architecture.

Success	Challenges
Program can be run from a Linux Desktop computer and an ARM microcontroller	
Vision algorithms are performed with 0% error margin compared with their Matlab counterpart. Speed increase is 650% on average	Segmentation of transparent particles is a challenge. This is critical to overcome, since systematic exclusion of a certain subset of particles gives inaccurate statistical results
Statistical calculations are performed within an error margin of 0.0001% compared with their Matlab counterparts. The speed increase is 400%	Since this class is used throughout the project and still takes up to 9% of the total time be called upon. Further optimization is advised.
Fast Fourier Transformation are performed within an error margin of 0.001% compared with their Matlab counterparts. The speed increase is 230%.	Further optimization is needed. The current C++ code is compiled with X64 desktop optimization. ARM machine code can perform this process with less instructions. Rewrite the FFT function in assembler for a big speed increase
The Neural Net can learning multiple Logical AND OR NAND setups.	Creation of an accurately classified soil particle database to learn the neural net.
	Overlap of smaller particles by bigger particles. The combination of the two samples overlapping particles where registered as one bigger particle. This gives distorted PSD results
	Scaling of pixels to SI unit mm
Light conditions case results in better reproducible color test results. Error margin of 10%.	Creation of a better sample environment
PCB electronics interfaces correctly with the ARM microcontroller	



Datum
5 juni 2015

Reference
xxxxxxxx

Version
Rev. A

Pagina
6 van 7



Figure 3 Embedded Microcontroller

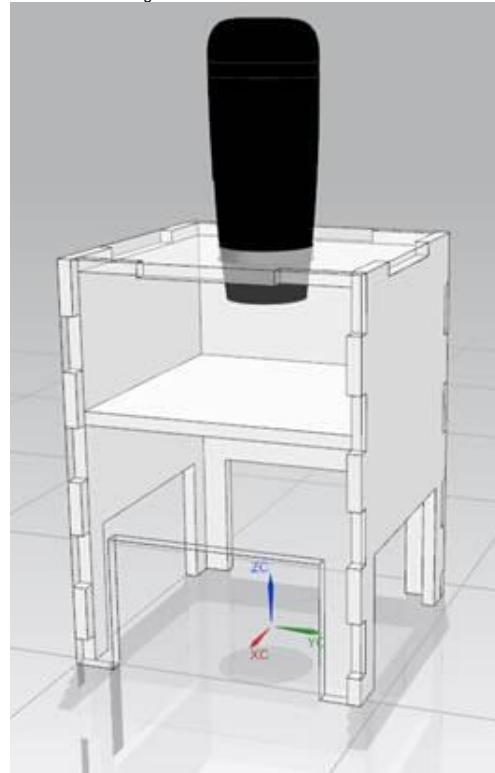


Figure 4 Light Environment Casing with Microscope

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

The technology innovator.



Datum
5 juni 2015

Reference
xxxxxxxx

Version
Rev. A

Pagina
7 van 7

4 Release candidate

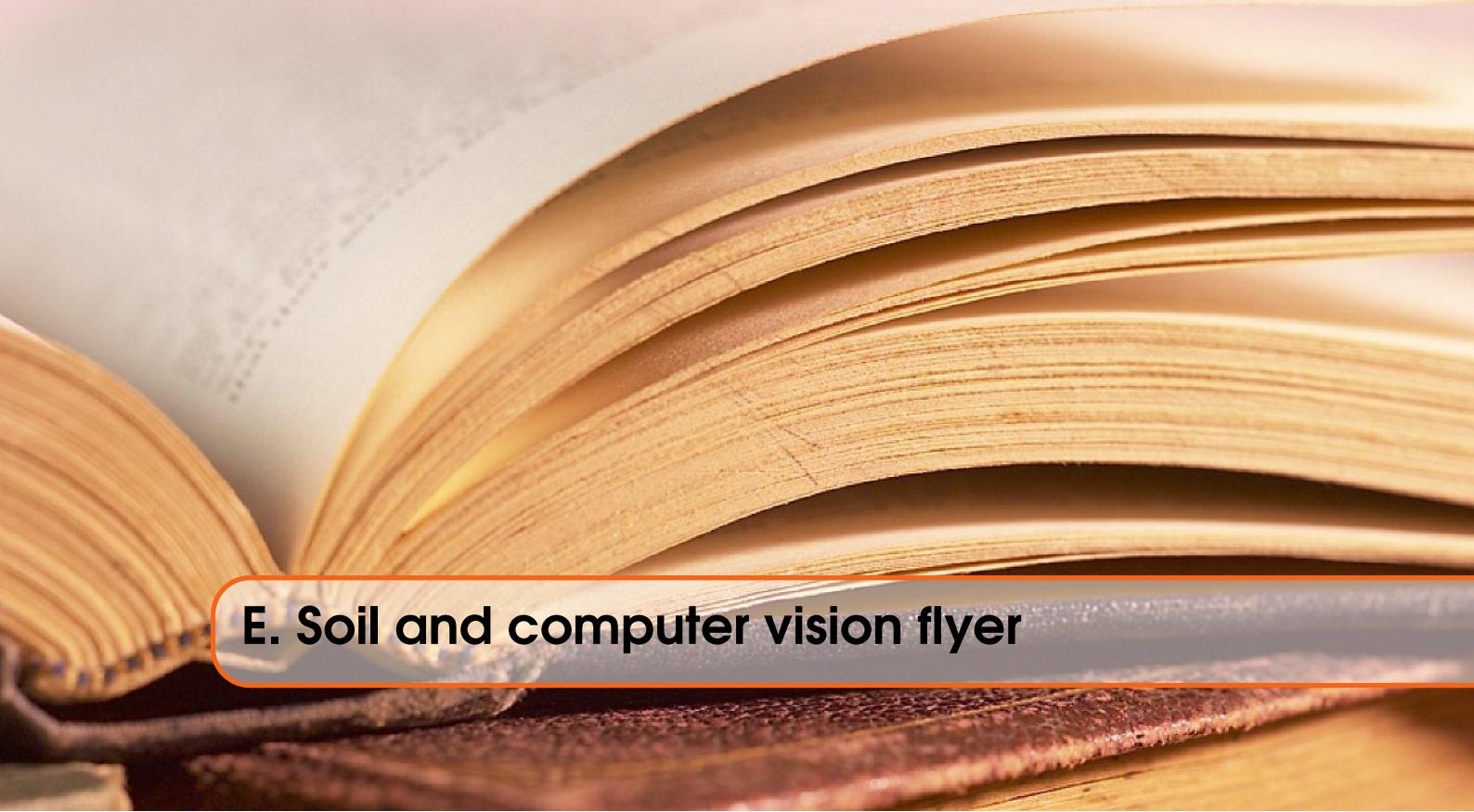
The goal of the future release candidate is to have a field ready device which is portable. The results are presented to the user using a generic HDMI monitor or can be download from the device in PDF format. These results fall in to a predefined and for a user acceptable error margin. The preliminary requirement below are an indication of possible requirements for a release candidate and are still subject to chance.

Functional:

- Calculations are done in a time span of five minutes.
- Calculation are within an acceptable and predefined error margin
- Results of the Particle Size Distribution are conforming NEN and ISO norms, such as but limited to NEN-ISO 9276-1 till 6.
- The device weighs less than 10 kg.
- The device can be lifted and carried by an adult human.
- The device can be used on a table with an max. level offset of 5°.
- The device complies at least with IP54 specifications.
- The device works at temperatures, ranging between -10°C / 40°C.
- Light conditions under the microscope are controlled.
- Results can be shared with other user or send to centralized database for further analysis.

Fabrication:

- The firm- and software can be updated remotely.
- The firm- and software can be easily maintained and should be well documented.
- Standardized internal hardware components are preferred.
- The casing and the internal mounting system can be manufactured using prototyping techniques, such as laser cutting and 3D printing.
- Each individual part is dismountable using standardized tools, such as Philips or cross screwdrivers.
- Costs of the used materials will be as low as possible.
- The device can be made as a small series with a max. of 50 devices.
- Further development with upscaling fabrication numbers will be taken in to account.
- A cradle to crate philosophy will be used in design and fabrication.



E. Soil and computer vision flyer

Vision Soil Analyzer

Current status and fare sight

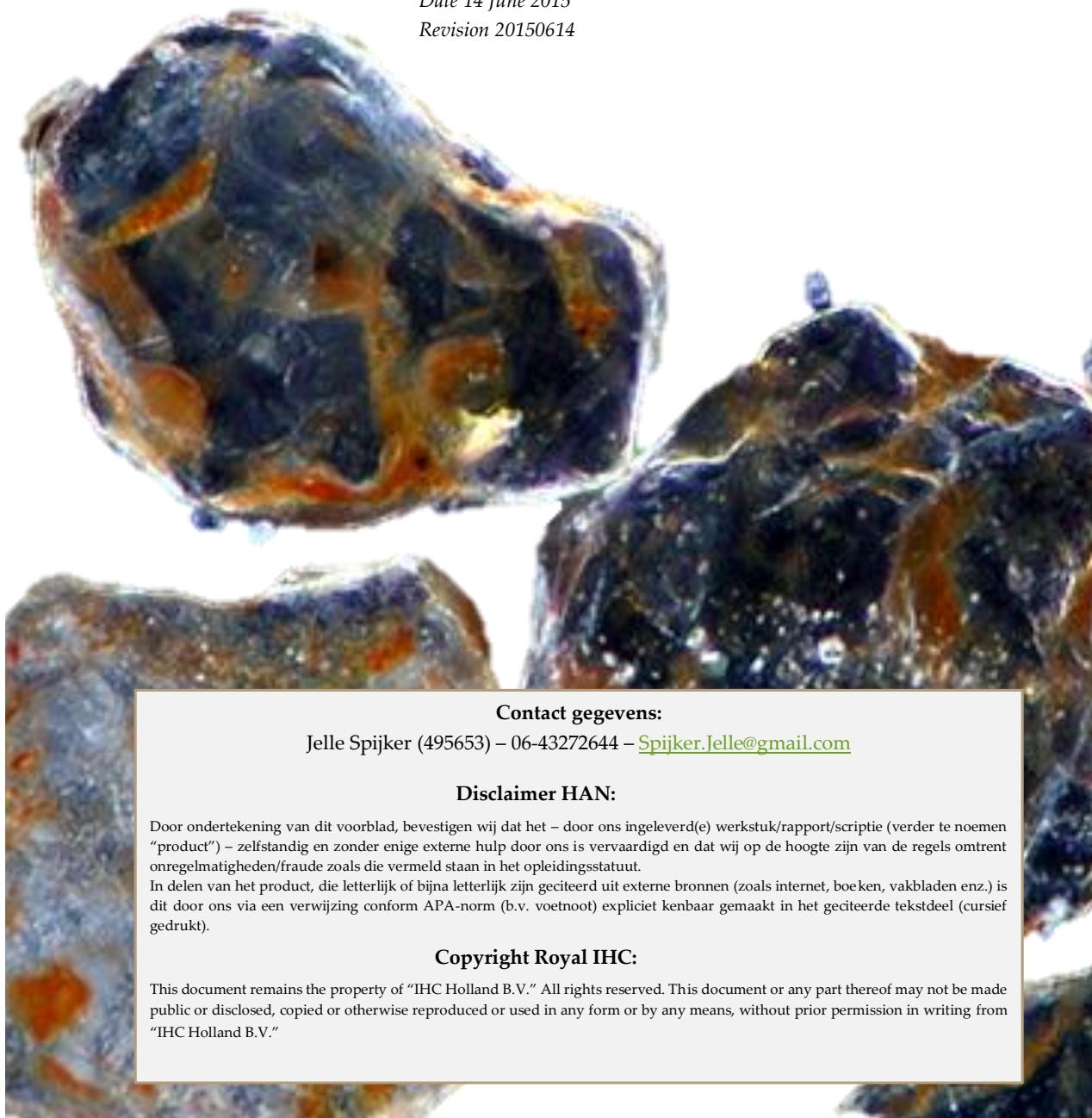
Client: Royal IHC – MTI Holland

Completion date: 15 juni 2015

Jelle Spijker

Date 14 June 2015

Revision 20150614



Contact gegevens:

Jelle Spijker (495653) – 06-43272644 – Spijker.Jelle@gmail.com

Disclaimer HAN:

Door ondertekening van dit voorblad, bevestigen wij dat het – door ons ingeleverd(e) werkstuk/rapport/scriptie (verder te noemen “product”) – zelfstandig en zonder enige externe hulp door ons is vervaardigd en dat wij op de hoogte zijn van de regels omtrent onregelmatigheden/fraude zoals die vermeld staan in het opleidingsstatuut.

In delen van het product, die letterlijk of bijna letterlijk zijn geciteerd uit externe bronnen (zoals internet, boeken, vakbladen enz.) is dit door ons via een verwijzing conform APA-norm (b.v. voetnoot) explicet kenbaar gemaakt in het geciteerde tekstdeel (cursief gedrukt).

Copyright Royal IHC:

This document remains the property of “IHC Holland B.V.” All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from “IHC Holland B.V.”

**IHC Holland B.V.**

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

TABLE OF CONTENT

TABLE OF CONTENT	2
1 INTRODUCTION.....	3
2 WORKING PRINCIPLE.....	5
3 VISION SOIL ANALYZER ALPHA.....	7
3.1 TECHNICAL SPECIFICATION	7
3.2 CHALLENGES TO OVERCOME.....	7
4 VISION SOIL ANALYZER BETA	8
4.1 TECHNICAL SPECIFICATION	8
4.2 CHALLENGES TO OVERCOME.....	8
5 THE RELEASE CANDIDATE.....	9
5.1 PRELIMINARY REQUIREMENTS.....	9
5.2 PROPOSED PROJECT SETUP	10
6 ROADMAP TO THE FUTURE.....	12
7 CONCLUSION.....	13
APPENDIX I. LITERATURE STUDY – SOIL AND IT'S OPTICAL PROPERTIES.....	14

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 2/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

1 Introduction

BACKGROUND

This project finds its roots in the minor Embedded Vision Design taught at the university of applied sciences HAN, hereafter named EVD. During this minor an embedded device was developed which analyses soil samples using a microscope. This Vision Soil Analyzer hereafter referred to as VSA, analyzes samples using the optical properties. It's main function is: *Presenting quantifiable information to a user on the properties of soil such as colour, texture and structure.*

IHC & MTI

This device is developed in collaboration with Royal IHC and MTI Holland. Royal IHC is one of Holland major shipyard companies and specializes in dredging and offshore. MTI Holland BV is royal IHC dredging knowledge center. They're worldwide leading center of expertise in the area of translating knowledge of dredging, mining and deep-sea mining processes into the specification, design and application of equipment. Both companies have an interests in knowing the properties of soil, be it to advise their customers or to further facilitate their own research and services.

ANALYZED PROPERTIES

The properties that can be analyzed using a digital camera and the preferred methods where determined by investigating the current literature, regarding soil, computer vision and various algorithms needed to perform the calculations. The detailed study can be found at appendix I.

TIME CONSUMING ARCHAIC METHODS

Current methods, like the Particle Size Analysis using a sieve and hydrometer are time consuming and non-portable. To facilitate quick, accurate and on location soil research an embedded device has been developed. This VSA analyzes soil samples using a microscope and gives the user acceptable and quick results on the soil visual properties.

QUICK & ACCURATE

Quick and reliable results are a welcome addition into any laboratory, this combined with a device that is light and portable gives its users an added benefit of shortened logistical operations for their soil samples. This results in some serious time benefits.

CURRENT & FUTURE PLANS

The current beta project brought a couple of complications to light which should be overcome in order to have a working release candidate. The release candidate will be further developed with the help of MTI and RDM campus and various study groups. The release candidate will be separated into a smaller subset of engineering and project challenges which will result in a working release candidate.

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 3/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.

**IHC Holland B.V.**

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

This document has the following structure. At first the basic working principle of the device in general is explained in chapter two. Secondly the alpha stage is briefly described in chapter three. Thirdly the beta and current stage is described, indicating which problems are still to overcome for the release candidate. This can be found in chapter four. In chapter five the release candidate is described. Naming the preliminary requirements and describing the proposed project setup. In chapter six there will be a short roadmap to the future. Finally the conclusion is drawn in chapter seven.

DOCUMENT STRUCTURE

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 4/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

2 Working principle

GOAL

As stated in the introduction, the goal is to develop a device which analyzes Soil samples using a digital microscopic camera connected to a microcontroller. The properties that are deemed possible to analyze using this technique are, color, texture and structure. The goal is to perform the calculation within a time span of a minute.

ANALYZED PROPERTIES

Color

- CIE La*b* color model presented as scatterplot with the mean values of each individual particle set out against the chromatic a* and b* axis. Studies indicate a correlation between organic carbon and the values in CIE La*b* color model
- Redness Index is presented as statistical data for each individual particle, such as mean, min, max, range, standard deviation etc. Welch tests anova can be executed in order to determine which particle deviates from the rest

Texture

- Particle Size Distribution Presented as a cumulative function. These properties show a correlation on water infiltration, pH buffering, buffering of organic materials and much more.

Structure

- Shape classification regarding each individual particle presented as histogram. The roundness and the angularity are determined and presented as sixteen individual classes. Ranging from high sphericity / well rounded to low sphericity / very angular. These properties show a correlation between erosion, biochemical and physical properties including tool degradation.

DEVICE DESCRIPTION

The program runs on an embedded Linux device. The algorithms which translate a digital snapshot of the magnified soil sample to the user preferred information are written in C++. They're by my own design and optimized to run on ARMv7 device running embed Linux. These algorithms can also be run from an X64 desktop computer, albeit optimization is not done for this environment. The performance is still better, because the average desktop computer has more system resources available.

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 5/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

Analyzing of the soil sample is done using the following workflow:

The soil sample is dried and the user makes sure the particle don't bond together. A small portion of the sample is placed on a sample plate. Taking care to separate the individual particles as much as possible. The cover is closed and a microscopic camera is positioned, in an environment where the light conditions are controlled.

The embedded Linux device takes a snapshot which is analyzed using the following computer algorithms:

First the individual soil particles are identified in the image, using various algorithms, such as adaptive contrast stretch, Gaussian blurring, OTSU – optimal thresholds separation.

The color information is determined with various matrix calculations, translating the RGB pixel value to CIE La*b* and Redness Index.

The texture information is determined by counting the number of discrete pixels for each individual particle. From this the volume is determined. If the scale of each pixel is known, the volume can be given in SI units.

The structure of an individual particle is determined by getting the edge of the pixels. This is done by creating a mask with a morphological erosion algorithm this mask is subtracted of the original image. The contour is translated to a function using the Dijkstra shortest path algorithm. Where each pixel is described as an imaginary complex number representing the radius towards the center of the particle. The vector holding these values are transformed to the frequency space using the Fast Fourier Transformation. The describing complex numbers gained during this transformation are fed into a feedforward Neural Network, which is optimized using Genetic Algorithms and a previously determined learning data set. The output is presented as probability that a certain particle belongs to a predefined category.

The results are presented to the user via a graphical user interface which are shown when the device is hooked to a monitor carrying a HDMI input. It's also possible to present a report in pdf or a native format which can be downloaded from the device using a LAN network device or optional Wi-Fi or Bluetooth. Basic human interaction can be performed via an onboard encoder, or optional USB keyboard and/or mouse.

WORKFLOW

PARTICLE SEGMENTATION

COLOR

TEXTURE

STRUCTURE

USER INTERFACE

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 6/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

3 Vision Soil Analyzer Alpha

This device was developed during the first part of the minor EVD. It served as a testing ground for the various algorithms. The goal was to develop a test setup with which to test the various computer algorithms and validate the theory.

3.1 Technical Specification

Hardware environment

- Pentium i3 2.3Ghz 4 cores
- 8 gb DDR2 memory
- Radeon HD mobility video card with 512gb dedicated memory
- SSD 128GB hard disk
- Generic 5 mp microscopic USB camera capable of 300X optical zoom

Software environment

- Windows 8.1
- Matlab 2014a

3.2 Challenges to overcome

- Calculations take a long time
 - Overcome by changing the programming language from Matlab to C++.
- Changing light conditions
 - Created a light condition environment.
- Segmentation of individual particle from the background
 - Partly overcome with vision enhancement algorithms
 - Still problems with translucent particles
- Overlap of smaller particles with bigger particles
 - By manual changing the configuration and analyze the soil sample multiple times a more accurate results can be determined.
- No learning data set available to feed the neural net with known structure values.
 - No obvious correlation could be determined.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

4 Vision Soil Analyzer Beta

During this project phase the switch was made from Matlab on a desktop environment to an embedded ARMv7 device running Linux. The algorithms are translated to the OOP – Object Orientated Programming language C++. Which is known for its speed and efficient handling of memory resources. Since each image consists of a matrix with around 5 million pixels in three color values. C++ was the logical choice. Due to the scale of the project (7000 lines of source and counting) this phase is still ongoing.

4.1 Technical Specification

Development environment

- Desktop computer running a Debian based Linux distribution
- Qt Creator IDE
- Various debugging tools

Runtime device

- ARMv7 based device Beaglebone black
- Generic USB microscopic digital camera
- Linux Ubuntu
- Various C++ development libraries
- Various Linux tools to facilitate optional specifications

4.2 Challenges to overcome

- Interfacing the camera with USB mode
 - Beaglebone specific problem. Overcome by reading the data from the webcam serial instead of parallel.
- Segmentation of individual particle from the background
 - Partly overcome with vision enhancement algorithms
 - Still problems with translucent particles
- Overlap of smaller particles with bigger particles
 - By manual changing the configuration and analyze the soil sample multiple times a more accurate results can be determined.
- No learning data set available to feed the neural net with known structure values.
 - Testing and learning of the neural net will be performed using synthetic pictures.

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 8/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

5 The release candidate

GOAL

The goal of this release candidate is to have a field ready device which is portable. The results are presented to the user using a generic HDMI monitor or can be download from the device in PDF format. These results fall in to a predefined and for a user acceptable error margin.

5.1 Preliminary requirements

Since this project phase has yet to commence the requirements below are preliminary and subject to chance.

Functional:

- Calculations are done in a time span of five minutes.
- Calculation are within an acceptable and predefined error margin
- Results of the Particle Size Distribution are conforming NEN and ISO norms, such as but limited to NEN-ISO 9276-1 till 6.
- The device weighs less than 10 kg.
- The device can be lifted and carried by an adult human.
- The device can be used on a table with an max. level offset of 5°.
- The device complies at least with IP54 specifications.
- The device works at temperatures, ranging between -10°C / 40°C.
- Light conditions under the microscope are controlled.
- Results can be shared with other user or send to centralized database for further analysis.

FABRICATION REQ.

Fabrication:

- The firm- and software can be updated remotely.
- The firm- and software can be easily maintained and should be well documented.
- Standardized internal hardware components are preferred.
- The casing and the internal mounting system can be manufactured using prototyping techniques, such as laser cutting and 3D printing.
- Each individual part is dismountable using standardized tools, such as Philips or cross screwdrivers.
- Costs of the used materials will be as low as possible.
- The device can be made as a small series with a max. of 50 devices.
- Further development with upscaling fabrication numbers will be taken in to account.
- A cradle to crate philosophy will be used in design and fabrication.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

5.2 Proposed project setup

In order to have manageable amount of work the following division is proposed:

1. Project oversight WTB
 - o Compiling Technical Dossier for CE marking.
 - Machinery directive.
 - Determine harmonized norms.
 - Determining final requirements (market research, norms, laws).
 - Writing test protocols.
 - Analyzing report (FMEA, DFA, RPA, etc.).
 - Gathering test results.
 - Protection of knowledge (patents).
 - Collecting and filing of output of individual projects.
 - Writing user manuals
 - o Determine and ensuring implementation of solutions for the problem of:
 - Overlapping particles.
 - Segmentation of translucent particles.
 - o Determine camera to be used and light conditions.
 - o Determine technical specification for other individual projects.
 - o Guarding individual project results.
2. Updating software to work with new device. ESD OR ICA
 - o Document the source code.
 - o Rewrite source code to work with new peripheral.
 - o Write camera driver.
 - o Write light environment driver
 - o Write Sample plate driver
 - o Write unit tests
3. Development of the casing and internal mounting. IPO OR WTB
 - o Translate technical specifications to requirements.
 - o Determine the design in accordance with the requirements
 - o Draw the design output as mono drawings.
 - o Generate a BOM
 - o Manufacture a prototype of the device in accordance with requirements and drawings.

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 10/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

- WTB
4. Development of the sample plate.
 - o Translate technical specifications to requirements.
 - o Determine the design in accordance with the requirements
 - o Draw the design output as mono drawings.
 - o Write calculation report.
 - o Generate a BOM.
 - o Manufacture a prototype of the device in accordance with requirements and drawings.
 5. Development of PCB and interface with the peripheral.
 - o Translate technical specifications to requirements.
 - o Determine the design in accordance with requirements.
 - o Draw the PCB.
 - o Generate a BOM.
 - o Write calculation report.
 - o Manufacture a prototype of the dive in accordance with requirements and design.
- EI OR ESD

DIVISION OF LABOR

It is advised that project oversight and software updating are kept under supervision by the original process owner, Jelle Spijker. The project for the sample plate and the casing can be executed with the regular HRO (university of applied sciences Rotterdam) WTB students at the RDM campus. Development of the PCB can be performed via a third party or by electrical / embedded system design students from university of applied sciences HAN.



IHC Holland B.V.

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

6 Roadmap to the future

If results from the release candidate are successful, this can serve as a basis for a consumer ready device. A device which has a place in any soil related laboratory, whether it's a stationary lab, field tent or a trailing suction hopper dredger.

The next challenge will be create a device which has a professional look, robust inner working, is economical priced and which will be easily maintained and upgraded.

But the biggest asset will be that its part of the IoT internet of things. Data gathered from this device can be uploaded to a centralized server. Trends can be spotted using this data, helping geo-engineers with their work all over the world. Offsite help can be given to users with questions, who want advise. And finally due to the use of self-learning Neural Networks it can optimize its results, learning from different devices and users.

CONSUMER READY DEVICE

INTERNET OF THINGS

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 12/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.

**IHC Holland B.V.**

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

7 Conclusion

This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

The technology innovator.

Page 13/14
Revision 20150614
Soil and computer vision – Concept

**IHC Holland B.V.**

P.O. Box 1, 2960 AA Kinderdijk
Smitweg 6, 2961 AW Kinderdijk

T +31 786 91 09 11
F +31 786 91 38 66
info@ihcmerwede.com
www.ihcmerwede.com

Appendix I. Literature study – Soil and it's optical properties

See external document

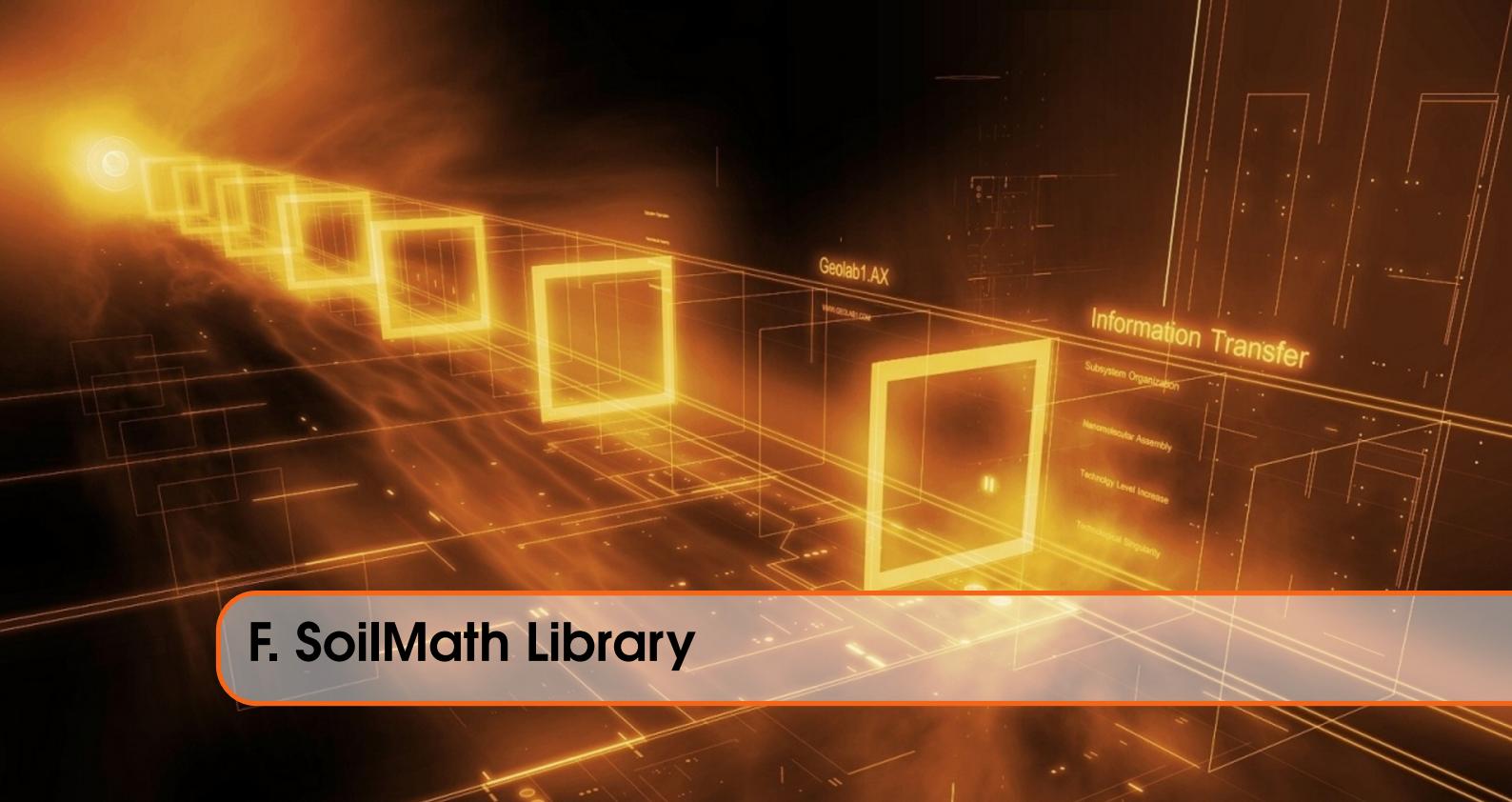
This document remains the property of "IHC Holland B.V." All rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from "IHC Holland B.V."

Page 14/14

Revision 20150614

Soil and computer vision – Concept

The technology innovator.



F. SoilMath Library

F.0.1 Genetic Algorithm Class

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10  /**
11   * Use this class for optimization problems. It's currently
12   * optimized for
13   * Neural Network optimzation
14   */
15 #pragma once
16
17 #include <bitset>
18 #include <random>
19 #include <string>
20 #include <algorithm>
21 #include <chrono>
22 #include <math.h>
23 #include <list>
24
25 // #include "NN.h"
26 #include "SoilMathTypes.h"
27 #include "MathException.h"
28 #include <QtCore/QObject>
29 #include <QDebug>
```

```
29 #include <QThread>
30 #include <QtConcurrent>
31
32 #include <boost/bind.hpp>
33
34 namespace SoilMath {
35
36 class GA : public QObject {
37     Q_OBJECT
38
39 public:
40     float MutationRate = 0.075f; /*< mutation rate*/
41     uint32_t Elitisme = 4;           /*< total number of the
42                                     elite bastard*/
43     float EndError = 0.001f;         /*< acceptable error between
44                                     last itteration*/
45     bool Revolution = true;
46
47     /*!
48     * \brief GA Standard constructor
49     */
50     GA();
51
52     /*!
53     * \brief GA Construction with a Neural Network
54     * initializers
55     * \param nnfunction the Neural Network prediction
56     * function which results will
57     * be optimized
58     * \param inputneurons the number of input neurons in the
59     * Neural Network don't
60     * count the bias
61     * \param hiddenneurons the number of hidden neurons in
62     * the Neural Network
63     * don't count the bias
64     * \param outputneurons the number of output neurons in
65     * the Neural Network
66     */
67     GA(NNfunctionType nnfunction, uint32_t inputneurons,
68         uint32_t hiddenneurons,
69         uint32_t outputneurons);
70
71     /*!
72     * \brief GA standard de constructor
73     */
74     ~GA();
75
76     /*!
77     * \brief Evolve Darwin would be proud!!! This function
78     * creates a population
79     * and itterates
80     * through the generation till the maximum number off
81     * itterations has been
82     * reached of the
83     * error is acceptable
84     */
85 }
```

```

74     * \param inputValues complex vector with a reference to
75     * the inputvalues
76     * \param weights reference to the vector of weights which
77     * will be optimized
78     * \param rangeweights reference to the range of weights,
79     * currently it doesn't
80     * support individul ranges
81     * this is because of the crossing
82     * \param goal target value towards the Neural Network
83     * prediction function
84     * will be optimized
85     * \param maxGenerations maximum number of itterations
86     * default value is 200
87     * \param popSize maximum number of population, this
88     * should be an even number
89     */
90     void Evolve(const InputLearnVector_t &inputValues,
91                 Weight_t &weights,
92                 MinMaxWeight_t rangeweights,
93                 OutputLearnVector_t &goal,
94                 uint32_t maxGenerations = 200, uint32_t
95                               popSize = 30);
96
97     signals:
98     void learnErrorUpdate(double newError);
99
100    private:
101    NNfunctionType NNfuction; /*< The Neural Net work
102    function*/
103    uint32_t inputneurons;    /*< the total number of input
104    neurons*/
105    uint32_t hiddenneurons;  /*< the total number of hidden
106    neurons*/
107    uint32_t outputneurons;  /*< the total number of output
108    neurons*/
109
110    MinMaxWeight_t rangeweights;
111    InputLearnVector_t inputValues;
112    OutputLearnVector_t goal;
113
114    float minOptim = 0;
115    float maxOptim = 0;
116    uint32_t oldElit = 0;
117    float oldMutation = 0.;
118    std::list<double> last10Gen;
119    uint32_t currentGeneration = 0;
120    bool revolutionOngoing = false;
121
122    /*!
123     * \brief Genesis private function which is the spark of
124     * live, using a random
125     * seed
126     * \param weights a reference to the used Weight_t vector
127     * \param rangeweights pointer to the range of weights,
128     * currently it doesn't
129     * support individul ranges

```

```
114     * \param popSize maximum number of population, this
115     *      should be an even number
116     */
117 Population_t Genesis(const Weight_t &weights, uint32_t
118     popSize);
119 /**
120 * \brief CrossOver a private function where the partners
121 *      mate with each other
122 * The values or PopMember_t are expressed as bits or are
123 *      cut at the point
124 * Crossover
125 * the population members are paired with the nearest
126 *      neighbor and new members
127 * are
128 * created pairing the Genome_t of each other at the
129 *      crossover point.
130 * Afterwards all
131 * the top tiers partners are allowed to mate again.
132 * \param pop reference to the population
133 */
134 void CrossOver(Population_t &pop);
135 /**
136 * \brief Mutate a private function where individual bits
137 *      from the Genome_t
138 * are mutated
139 * at a random uniform distribution event defined by the
140 *      mutationrate
141 * \param pop reference to the population
142 */
143 void Mutate(Population_t &pop);
144 /**
145 * \brief GrowToAdulthood a private function where the new
146 *      population members
147 * serve as the
148 * the input for the Neural Network prediction function.
149 * The results are
150 * weight against
151 * the goal and this weight determine the fitness of the
152 *      population member
153 * \param pop reference to the population
154 * \param inputValues a InputLearnVector_t with a
155 *      reference to the inputvalues
156 * \param rangeweights pointer to the range of weights,
157 *      currently it doesn't
158 * support individual ranges
159 * \param goal a Predict_t type with the expected value
160 * \param totalFitness a reference to the total population
161 *      fitness
162 */
163 void GrowToAdulthood(Population_t &pop, float &
164     totalFitness);
```

```

155 	/*!
156 	* \brief SurvivalOfTheFittest a private function where a
157 	    battle to the death
158 	* commences
159 	* The fittest population members have the best chance of
160 	    survival. Death is
161 	* instigated
162 	* with a random uniform distribution. The elite members
163 	    don't partake in this
164 	* desctruction
165 	* The ELITISME rate indicate how many top tier members
166 	    survive this
167 	* catastrophic event.
168 	* \param inputValues a InputLearnVector_t with a
169 	    reference to the inputvalues
170 	* \param totalFitness a reference to the total population
171 	    fitness
172 	* \return
173 	* /
174 	bool SurvivalOfTheFittest(Population_t &pop, float &
175 	    totalFitness);
176
177 	/*!
178 	* \brief PopMemberSort a private function where the
179 	    members are sorted
180 	* according to
181 	* there fitness ranking
182 	* \param i left hand population member
183 	* \param j right hand population member
184 	* \return true if the left member is closer to the goal
185 	    as the right member.
186 	* /
187 	static bool PopMemberSort(PopMember_t i, PopMember_t j) {
188 	    return (i.Fitness < j.Fitness);
189 	}
190
191 	/*!
192 	* \brief Conversion of the value of type T to Genome_t
193 	* \details Usage: Use <tt>ConvertToGenome<Type>(type,
194 	    range)</tt>
195 	* \param value The current value which should be converted
196 	    to a Genome_t
197 	* \param range the range in which the value should fall,
198 	    this is to have a
199 	* Genome_t
200 	* which utilizes the complete range 0000...n till 1111...
201 	    n
202 	* /
203 	template <typename T>
204 	inline Genome_t ConvertToGenome(T value, std::pair<T, T>
205 	    range) {
206 	    uint32_t intVal = static_cast<uint32_t>(
207         (UINT32_MAX * (range.first + value)) / (range.second
208             - range.first));
209 	    Genome_t retVal(intVal);
210 	    return retVal;

```

```

196     }
197
198     /*!
199      * \brief Conversion of the Genome to a value
200      * \details Usage: use <tt>ConvertToValue<Type>(genome,
201      * range)
202      * \param gen is the Genome which is to be converted
203      * \param range is the range in which the value should
204      * fall
205      */
206     template <typename T>
207     inline T ConvertToValue(Genome_t gen, std::pair<T, T>
208     range) {
209     T retVal =
210         range.first +
211         (((range.second - range.first) * static_cast<T>(gen.
212             to_ulong()))) /
213         (UINT32_MAX);
214     return retVal;
215 }
216 };
217 }
218 }



---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8  */
9
8 #include "GA.h"
9
10 namespace SoilMath {
11 GA::GA() {}
12
13 GA::GA(NNfunctionType nnfunction, uint32_t inputneurons,
14         uint32_t hiddenneurons,
15         uint32_t outputneurons) {
16     this->NNfunction = nnfunction;
17     this->inputneurons = inputneurons;
18     this->hiddenneurons = hiddenneurons;
19     this->outputneurons = outputneurons;
20 }
21
21 GA::~GA() {}
22
23 void GA::Evolve(const InputLearnVector_t &inputValues,
24                   Weight_t &weights,
25                   MinMaxWeight_t rangeweights,
26                   OutputLearnVector_t &goal,
27                   uint32_t maxGenerations, uint32_t popSize) {
28     minOptim = goal[0].OutputNeurons.size();
29     minOptim = -minOptim;
30     maxOptim = 2 * goal[0].OutputNeurons.size();

```

```

29     oldElit = Elitisme;
30     oldMutation = MutationRate;
31     this->inputValues = inputValues;
32     this->rangeweights = rangeweights;
33     this->goal = goal;
34
35     // Create the population
36     Population_t pop = Genesis(weights, popSize);
37     float totalFitness = 0.0;
38     for (uint32_t i = 0; i < maxGenerations; i++) {
39         CrossOver(pop);
40         Mutate(pop);
41         totalFitness = 0.0;
42         GrowToAdulthood(pop, totalFitness);
43         if (SurvivalOfTheFittest(pop, totalFitness)) {
44             break;
45         }
46     }
47     weights = pop[0].weights;
48 }
49
50 Population_t GA::Genesis(const Weight_t &weights, uint32_t
51     popSize) {
52     if (popSize < 1)
53         return Population_t();
54
55     Population_t pop;
56     unsigned seed = std::chrono::system_clock::now().
57         time_since_epoch().count();
58     std::default_random_engine gen(seed);
59     std::uniform_real_distribution<float> dis(rangeweights.
60         first,
61                                         rangeweights.
62                                         second);
63
64     for (uint32_t i = 0; i < popSize; i++) {
65         PopMember_t I;
66         for (uint32_t j = 0; j < weights.size(); j++) {
67             I.weights.push_back(dis(gen));
68             I.weightsGen.push_back(
69                 ConvertToGenome<float>(I.weights[j], rangeweights)
70             );
71         }
72         pop.push_back(I);
73     }
74     return pop;
75 }
76
77 void GA::CrossOver(Population_t &pop) {
78     Population_t newPop; // create a new population
79     PopMember_t newPopMembers[2];
80     SplitGenome_t Split[2];
81
82     for (uint32_t i = 0; i < pop.size(); i += 2) {
83         for (uint32_t j = 0; j < pop[i].weights.size(); j++) {

```

```

Crossover ,
120                                     GENE_MAX
-
Crossover
));

121
122     newPopMembers[0].weightsGen.push_back(
123         Genome_t(Split[0].first.to_string() + Split[1].
124             second.to_string()));
125     newPopMembers[1].weightsGen.push_back(
126         Genome_t(Split[1].first.to_string() + Split[0].
127             second.to_string()));
128
129     newPop.push_back(newPopMembers[0]);
130     newPop.push_back(newPopMembers[1]);
131     newPopMembers[0].weightsGen.clear();
132     newPopMembers[1].weightsGen.clear();
133 }
134
135 void GA::Mutate(Population_t &pop) {
136     unsigned seed = std::chrono::system_clock::now().
137         time_since_epoch().count();
138     std::default_random_engine gen(seed);
139     std::uniform_real_distribution<float> dis(0, 1);
140
141     std::default_random_engine genGen(seed);
142     std::uniform_int_distribution<int> disGen(0, (GENE_MAX -
143
144     QtConcurrent::blockingMap<Population_t>(pop, [&]{
145         PopMember_t &P) {
146             for (uint32_t j = 0; j < P.weightsGen.size(); j++) {
147                 if (dis(gen) < MutationRate) {
148                     P.weightsGen[j][disGen(genGen)].flip();
149                 }
150             }
151         });
152
153     void GA::GrowToAdulthood(Population_t &pop, float &
154         totalFitness) {
155
156         QtConcurrent::blockingMap<Population_t>(pop, [&]{
157             PopMember_t &P) {
158                 // std::for_each(pop.begin(), pop.end(), [&](PopMember_t
159                 // &P) {
160                     for (uint32_t j = 0; j < P.weightsGen.size(); j++) {
161                         P.weights.push_back(ConvertToValue<float>(P.weightsGen
162                             [j], rangeweights));
163                     }
164                     Weight_t iWeight(P.weights.begin(),
165                         P.weights.begin() + ((inputneurons + 1)
166                             * hiddenneurons));
167                     Weight_t hWeight(P.weights.begin() + ((inputneurons + 1)
168

```

```

162         * hiddenneurons),
163         P.weights.end());
164
165     for (uint32_t j = 0; j < inputValues.size(); j++) {
166         Predict_t results = NNfuction(inputValues[j], iWeight,
167                                         hWeight,
168                                         inputneurons,
169                                         hiddenneurons,
170                                         outputneurons);
171
172         // See issue #85
173         bool allGood = true;
174         float fitness = 0.0;
175         for (uint32_t k = 0; k < results.OutputNeurons.size();
176              k++) {
177             bool resultSign = std::signbit(results.OutputNeurons
178                                           [k]);
179             bool goalSign = std::signbit(goal[j].OutputNeurons[k
180                                           ]);
181             fitness += results.OutputNeurons[k] / goal[j].
182                         OutputNeurons[k];
183             if (resultSign != goalSign) {
184                 allGood = false;
185             }
186         }
187         fitness += (allGood) ? results.OutputNeurons.size() :
188                         0;
189         P.Fitness += fitness;
190     }
191 }
192
193 for_each(pop.begin(), pop.end(), [&](PopMember_t &P) {
194     P.Fitness /= inputValues.size();
195     totalFitness += P.Fitness;
196 });
197
198 bool GA::SurvivalOfTheFittest(Population_t &pop, float &
199                                 totalFitness) {
200     bool retVal = false;
201     uint32_t decimationCount = pop.size() / 2;
202
203     unsigned seed = std::chrono::system_clock::now().
204                     time_since_epoch().count();
205     std::default_random_engine gen(seed);
206
207     std::sort(pop.begin(), pop.end(),
208               [](&const PopMember_t &L, &const PopMember_t &R) {
209                 return L.Fitness < R.Fitness;
210             });
211
212     float maxFitness = pop[pop.size() - 1].Fitness * pop.size
213     ();
214     uint32_t i = Elitisme;
215     while (pop.size() > decimationCount) {
216         if (i == pop.size()) {
217             i = Elitisme;
218         }
219     }
220 }
```

```

206     }
207     std::uniform_real_distribution<float> dis(0, maxFitness)
208     ;
209     if (dis(gen) > pop[i].Fitness) {
210         totalFitness -= pop[i].Fitness;
211         pop.erase(pop.begin() + i);
212     }
213     i++;
214 }
215 std::sort(pop.begin(), pop.end(),
216           [] (const PopMember_t &L, const PopMember_t &R) {
217               return L.Fitness > R.Fitness;
218           });
219
220 float learnError = 1 - ((pop[0].Fitness - minOptim) / (
221     maxOptim - minOptim));
222
223 // Viva la Revolution
224 if (currentGeneration > 9) {
225     double avg = 0;
226     for_each(last10Gen.begin(), last10Gen.end(), [&] (double
227         &G) { avg += G; });
228     avg /= 10;
229     double minMax[2] = {avg * 0.98, avg * 1.02};
230     if (learnError > minMax[0] && learnError < minMax[1]) {
231         if (!revolutionOngoing) {
232             qDebug() << "Viva la revolution!";
233             oldElit = Elitisme;
234             Elitisme = 0;
235             oldMutation = MutationRate;
236             MutationRate = 0.25;
237             revolutionOngoing = true;
238         } else if (revolutionOngoing) {
239             qDebug() << "Peace has been restart";
240             Elitisme = oldElit;
241             MutationRate = oldMutation;
242             revolutionOngoing = false;
243         }
244         last10Gen.pop_front();
245         last10Gen.push_back(learnError);
246     } else {
247         last10Gen.push_back(learnError);
248     }
249     currentGeneration++;
250     emit learnErrorUpdate(static_cast<double>(learnError));
251     if (learnError < EndError) {
252         retVal = true;
253     }
254 }
255 }
```

F.0.2 Fast Fourier Transform Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include <vector>
13 #include <complex>
14 #include <cmath>
15 #include <valarray>
16 #include <array>
17 #include <deque>
18 #include <queue>
19 #include <iterator>
20 #include <algorithm>
21 #include <stdint.h>
22 #include <opencv2/core.hpp>
23 #include "SoilMathTypes.h"
24 #include "MathException.h"
25
26 namespace SoilMath {
27 /*!
28  * \brief Fast Fourier Transform class
29  * \details Use this class to transform a black and white
30  * blob presented as a
31  * cv::Mat with values 0 or 1 to a vector of complex values
32  * representing the Fourier
33  * Descriptors.
34  */
35 class FFT {
36 public:
37 /*!
38  * \brief Standard constructor
39  */
40 FFT();
41
42 /*!
43  * \brief Standard deconstructor
44  */
45 ~FFT();
46
47 /*!
48  * \brief Transforming the img to the frequency domain and
49  * returning the
50  * Fourier Descriptors
51  * \param img contour in the form of a cv::Mat type
52  * CV_8UC1. Which should
53  * consist of a continous contour. \f$ \{ img \in \mathbb{Z} \} | 0 \leq img \leq

```

```

48     * 1 \} \f$ 
49     * \return a vector with complex values, represing the
50     * contour in the
51     * frequency domain, expressed as Fourier Descriptors
52     */
53     ComplexVect_t GetDescriptors(const cv::Mat &img);
54
55 private:
56     ComplexVect_t
57     fftDescriptors; /*< Vector with complex values which
58     * represent the
59     * descriptors*/
60     ComplexVect_t
61     complexcontour; /*< Vector with complex values which
62     * represent the
63     * contour*/
64     cv::Mat Img;           /*< Img which will be analysed*/
65
66 /*!
67  * \brief Contour2Complex a private function which
68  * translates a continous
69  * contour image
70  * to a vector of complex values. The contour is found
71  * using a depth first
72  * search with
73  * extension list. The alghorithm is based upon <a
74  * href="http://ocw.mit.edu/courses/electrical-engineering-
75  * -and-computer-science/6-034-artificial-intelligence-
76  * -fall-2010/lecture-videos/lecture-4-search-depth-first-
77  * hill-climbing-beam/">MIT
78  * opencourseware
79  * 6-034-artificial-intelligence lecture 4</a>
80  * \param img contour in the form of a cv::Mat type
81  * CV_8UC1. Which should
82  * consist of a continous contour. \f$ \{ img \in \mathbb{Z} \mid 0 \leq img \leq
83  * 1 \} \f$ 
84  * \param centerCol centre of the contour X value
85  * \param centerRow centre of the contour Y value
86  * \return a vector with complex values, represing the
87  * contour as a function
88  */
89     ComplexVect_t Contour2Complex(const cv::Mat &img, float
90     centerCol,
91                                         float centerRow);
92
93 /*!
94  * \brief Neighbors a private function returning the
95  * neighboring pixels which
96  * belong to a contour
97  * \param 0 uchar pointer to the data
98  * \param pixel current counter
99  * \param columns total number of columns
100 * \param rows total number of rows
101 * \return
102 */

```

```

90     iContour_t Neighbors(uchar *0, int pixel, uint32_t columns
91                     , uint32_t rows);
92
93     /*!
94      * \brief fft a private function calculating the Fast
95      * Fourier Transform
96      * let  $m$  be an integer and let  $N=2^m$  also
97      *  $CA=[x_0, \dots, x_{N-1}]$  is an  $N$  dimensional
98      * dimensional complex vector
99      * let  $\omega = \exp\left(\frac{-2\pi i}{N}\right)$ 
100     * then  $c_k = \frac{1}{N} \sum_{j=0}^{N-1} CA_j \omega^{jk}$ 
101     * param CA a  $CA=[x_0, \dots, x_{N-1}]$  is an  $N$ 
102     * dimensional
103     * complex vector
104     */
105     void fft(ComplexArray_t &CA);
106
107     /*!
108      * \brief ifft
109      * \param CA
110      */
111     void ifft(ComplexArray_t &CA);
112
113 };
114 }

```

```

27     double nextLogN = floor(logN + 1.0);
28     N = static_cast<uint32_t>(pow(2, nextLogN));
29
30     uint32_t i = complexcontour.size();
31     // Append the vector with zeros
32     while (i++ < N) {
33         complexcontour.push_back(Complex_t(0.0, 0.0));
34     }
35 }
36
37 ComplexArray_t ca(complexcontour.data(), complexcontour.
38     size());
39 fft(ca);
40 fftDescriptors.assign(std::begin(ca), std::end(ca));
41 return fftDescriptors;
42
43 iContour_t FFT::Neighbors(uchar *0, int pixel, uint32_t
44     columns,
45     uint32_t rows) {
46     long int LUT_nBore[8] = {-columns + 1, -columns, -columns
47     - 1, -1,
48     columns - 1, columns, 1 +
49     columns, 1};
50     iContour_t neighbors;
51     uint32_t pEnd = rows * columns;
52     uint32_t count = 0;
53     for (uint32_t i = 0; i < 8; i++) {
54         count = pixel + LUT_nBore[i];
55         while (count >= pEnd && i < 8) {
56             count = pixel + LUT_nBore[++i];
57         }
58         if (i >= 8) {
59             break;
60         }
61         if (0[count] == 1)
62             neighbors.push_back(count);
63     }
64     return neighbors;
65 }
66
67 ComplexVect_t FFT::Contour2Complex(const cv::Mat &img, float
68     centerCol,
69     float centerRow) {
70     uchar *0 = img.data;
71     uint32_t pEnd = img.cols * img.rows;
72
73     std::deque<std::deque<uint32_t>> sCont;
74     std::deque<uint32_t> eList;
75
76     // Initialize the queue
77     for (uint32_t i = 0; i < pEnd; i++) {
78         if (0[i] == 1) {
79             std::deque<uint32_t> tmpQ;
80             tmpQ.push_back(i);
81             sCont.push_back(tmpQ);
82         }
83     }
84
85     while (!sCont.empty()) {
86         std::deque<uint32_t> &tmpQ = sCont.front();
87         uint32_t i = tmpQ.back();
88         sCont.pop_front();
89
90         for (int j = 0; j < 8; j++) {
91             int n = i + LUT_nBore[j];
92             if (n < 0 || n >= pEnd)
93                 continue;
94             if (0[n] == 1) {
95                 0[n] = 0;
96                 eList.push_back(n);
97             }
98         }
99     }
100
101    std::vector<Complex_t> complexContour;
102    for (uint32_t i = 0; i < eList.size(); i++) {
103        uint32_t n = eList[i];
104        complexContour.push_back(Complex_t(0.0, 0.0));
105        for (int j = 0; j < 8; j++) {
106            int n2 = n + LUT_nBore[j];
107            if (n2 < 0 || n2 >= pEnd)
108                continue;
109            if (0[n2] == 1) {
110                0[n2] = 0;
111                eList.push_back(n2);
112            }
113        }
114    }
115
116    ComplexArray_t ca(complexContour.data(), complexContour.
117        size());
118    fft(ca);
119    fftDescriptors.assign(std::begin(ca), std::end(ca));
120    return fftDescriptors;
121 }
```

```

78         break;
79     }
80 }
81
82 if (sCont.front().size() < 1) {
83     throw Exception::MathException(
84         EXCEPTION_NO_CONTOUR_FOUND,
85         EXCEPTION_NO_CONTOUR_FOUND_NR
86     );
87 } // Exception handling
88
89 uint32_t prev = -1;
90
91 // Extend path on queue
92 for (uint32_t i = sCont.front().front(); i < pEnd;) {
93     iContour_t nBors =
94         Neighbors(0, i, img.cols, img.rows); // find
95         neighboring pixels
96     std::deque<uint32_t> cQ = sCont.front(); // store first
97         queue;
98     sCont.erase(sCont.begin()); // erase first
99         queue from beginning
100    if (cQ.size() > 1) {
101        prev = cQ.size() - 2;
102    } else {
103        prev = 0;
104    }
105    // Loop through each neighbor
106    for (uint32_t j = 0; j < nBors.size(); j++) {
107        if (nBors[j] != cQ[prev]) // No backtracking
108        {
109            if (nBors[j] == cQ.front() && cQ.size() > 8) {
110                i = pEnd;
111            } // Back at first node
112            if (std::find(eList.begin(), eList.end(), nBors[j]) ==
113                eList.end()) // Check if this current route is
114                extended elsewhere
115            {
116                std::deque<uint32_t> nQ = cQ;
117                nQ.push_back(nBors[j]); // Add the neighbor to the
118                queue
119                sCont.push_front(nQ); // add the sequence to the
120                front of the queue
121            }
122        }
123    }
124 }
125
126 if (nBors.size() > 2) {
127     eList.push_back(i);
128 } // if there are multiple choices put current node in
129     extension List
130 if (i != pEnd) {
131     i = sCont.front().back();
132 } // If it isn't the end set i to the last node of the
133     first queue
134 if (sCont.size() == 0) {

```

```

123     throw Exception::MathException(
124         EXCEPTION_NO_CONTOUR_FOUND,
125         EXCEPTION_NO_CONTOUR_FOUND_NR
126     );
127 }
128 // convert the first queue to a complex normalized vector
129 Complex_t cPoint;
130 ComplexVect_t contour;
131 float col = 0.0;
132 // Normalize and convert the complex function
133 for_each(
134     sCont.front().begin(), sCont.front().end(),
135     [&img, &cPoint, &contour, &centerCol, &centerRow, &col
136     ](uint32_t &e) {
137     col = (float)((e % img.cols) - centerCol);
138     if (col == 0.0) {
139         cPoint.real(1.0);
140     } else {
141         cPoint.real((float)(col / centerCol));
142     }
143     cPoint.imag((float)((floord(e / img.cols) -
144         centerRow) / centerRow));
145     contour.push_back(cPoint);
146 });
147 }
148
149 void FFT::fft(ComplexArray_t &CA) {
150     const size_t N = CA.size();
151     if (N <= 1) {
152         return;
153     }
154
155     //!< Divide and conquer
156     ComplexArray_t even = CA[std::slice(0, N / 2, 2)];
157     ComplexArray_t odd = CA[std::slice(1, N / 2, 2)];
158
159     fft(even);
160     fft(odd);
161
162     for (size_t k = 0; k < N / 2; ++k) {
163         Complex_t ct = std::polar(1.0, -2 * M_PI * k / N) * odd[
164             k];
165         CA[k] = even[k] + ct;
166         CA[k + N / 2] = even[k] - ct;
167     }
168 }
169 void FFT::ifft(ComplexArray_t &CA) {
170     CA = CA.apply(std::conj);
171     fft(CA);
172     CA = CA.apply(std::conj);
173     CA /= CA.size();

```

174 }

175 }

F.0.3 Neural Network Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include <stdint.h>
13 #include <vector>
14 #include <string>
15 #include <fstream>
16
17 #include <boost/archive/xml_iarchive.hpp>
18 #include <boost/archive/xml_oarchive.hpp>
19 #include <boost/serialization/vector.hpp>
20 #include <boost/serialization/version.hpp>
21
22 #include "GA.h"
23 #include "MathException.h"
24 #include "SoilMathTypes.h"
25 #include "FFT.h"
26
27 #include <QtCore/QObject>
28
29 namespace SoilMath {
30 /*!
31  * \brief The Neural Network class
32  * \details This class is used to make prediction on large
33  * data set. Using self
34  * learning algoritmes
35  */
36 class NN : public QObject {
37     Q_OBJECT
38
39 public:
40     /*!
41      * \brief NN constructor for the Neural Net
42      * \param inputneurons number of input neurons
43      * \param hiddenneurons number of hidden neurons
44      * \param outputneurons number of output neurons
45      */
46     NN(uint32_t inputneurons, uint32_t hiddenneurons, uint32_t
47         outputneurons);
48
49     /*!
50      * \brief NN constructor for the Neural Net
51      */
52     NN();
53
54     /*!

```

```

51  * \brief ~NN virtual deconstructor for the Neural Net
52  */
53  virtual ~NN();
54
55 /*!
56  * \brief Predict The prediction function.
57  * \details In this function the neural net is setup and
58  * the input which are
59  * the complex values descriping the contour in the
60  * frequency domein serve as
61  * input. The absolute value of these im. number because I
62  * 'm not interrested
63  * in the orrientation of the particle but more in the
64  * degree of variations.
65  * \param input vector of complex input values, these're
66  * the Fourier
67  * descriptors
68  * \return a real valued vector of the output neurons
69  */
70 Predict_t Predict(ComplexVect_t input);
71
72 /*!
73  * \brief PredictLearn a static function used in learning
74  * of the weights
75  * \details It starts a new Neural Network object and
76  * passes all the
77  * paramaters in to this newly created object. After this
78  * the predict function
79  * is called and the value is returned. This work around
80  * was needed to pass
81  * the neural network to the Genetic Algorithm class.
82  * \param input a complex vector of input values
83  * \param inputweights the input weights
84  * \param hiddenweights the hidden weights
85  * \param inputneurons the input neurons
86  * \param hiddenneurons the hidden neurons
87  * \param outputneurons the output neurons
88  * \return
89  */
90 static Predict_t PredictLearn(ComplexVect_t input,
91     Weight_t inputweights,
92     Weight_t hiddenweights,
93     uint32_t inputneurons,
94     uint32_t hiddenneurons,
95     uint32_t outputneurons);
96
97 /*!
98  * \brief SetInputWeights a function to set the input
99  * weights
100 * \param value the real valued vector with the values
101 */
102 void SetInputWeights(Weight_t value) { iWeights = value; }
103
104 /*!
105  * \brief SetHiddenWeights a function to set the hidden
106  * weights

```

```

93     * \param value the real valued vector with the values
94     */
95     void SetHiddenWeights(Weight_t value) { hWeights = value;
96     }
97 /**
98     * \brief SetBeta a function to set the beta value
99     * \param value a floating value usually between 0.5 and
100    1.5
101   */
102  void SetBeta(float value) { beta = value; }
103  float GetBeta() { return beta; }
104 /**
105  * \brief Learn the learning function
106  * \param input a vector of vectors with complex input
107  values
108  * \param cat a vector of vectors with the know output
109  values
110  * \param noOfDescriptorsUsed the total number of
111  descriptors which should be
112  * used
113  */
114  void Learn(InputLearnVector_t input, OutputLearnVector_t
115  cat,
116  uint32_t noOfDescriptorsUsed);
117 /**
118  * \brief SaveState Serialize and save the values of the
119  Neural Net to disk
120  * \details Save the Neural Net in XML valued text file to
121  disk so that a
122  * object can
123  * be reconstructed on a latter stadia.
124  * \param filename a string indicating the file location
125  and name
126  */
127  void SaveState(std::string filename);
128 /**
129  * \brief LoadState Loads the previous saved Neural Net
130  from disk
131  * \param filename a string indicating the file location
132  and name
133  */
134  void LoadState(std::string filename);
135
136 Weight_t iWeights; /**< a vector of real valued floating
137  point input weights*/
138 Weight_t hWeights; /**< a vector of real valued floating
139  point hidden weight*/
140
141 uint32_t MaxGenUsedByGA = 200;
142 uint32_t PopulationSizeUsedByGA = 30;
143 float MutationrateUsedByGA = 0.075f;
144 uint32_t ElitismeUsedByGA = 4;

```

```
136     float EndErrorUsedByGA = 0.001;
137     float MaxWeightUsedByGA = 50;
138     float MinWeightUsedByGA = -50;
139
140     uint32_t GetInputNeurons() { return inputNeurons; }
141     void SetInputNeurons(uint32_t value);
142
143     uint32_t GetHiddenNeurons() { return hiddenNeurons; }
144     void SetHiddenNeurons(uint32_t value);
145
146     uint32_t GetOutputNeurons() { return outputNeurons; }
147     void SetOutputNeurons(uint32_t value);
148
149     bool studied =
150         false; /*< a value indicating if the weights are a
151             results of a
152                 learning curve*/
153
154     signals:
155         void learnErrorUpdate(double newError);
156
157     private:
158         GA *optim = nullptr;
159         std::vector<float> iNeurons; /*< a vector of input values
160             , the bias is
161                 included, the bias is
162                     included and
163                         is the first value*/
164         std::vector<float>
165             hNeurons; /*< a vector of hidden values, the bias is
166                 included and
167                     is the first value*/
168         std::vector<float> oNeurons; /*< a vector of output
169             values*/
170
171         uint32_t hiddenNeurons = 50; /*< number of hidden neurons
172             minus bias*/
173         uint32_t inputNeurons = 20; /*< number of input neurons
174             minus bias*/
175         uint32_t outputNeurons = 18; /*< number of output neurons
176             */
177         float beta; /*< the beta value, this indicates the
178             steepness of the sigmoid
179                 function*/
180
181     friend class boost::serialization::access; /*< a private
182             friend class so the
183                 serialization
184                     can
185                         access
186                             all
187                                 the needed
188                                     functions
189                                     */
190
191     /*!
192     * \brief serialization function
```

```

177     * \param ar the object
178     * \param version the version of the class
179     */
180     template <class Archive>
181     void serialize(Archive &ar, const unsigned int version) {
182         if (version == 0) {
183             ar &BOOST_SERIALIZATION_NVP(inputNeurons);
184             ar &BOOST_SERIALIZATION_NVP(hiddenNeurons);
185             ar &BOOST_SERIALIZATION_NVP(outputNeurons);
186             ar &BOOST_SERIALIZATION_NVP(iWeights);
187             ar &BOOST_SERIALIZATION_NVP(hWeights);
188             ar &BOOST_SERIALIZATION_NVP(beta);
189             ar &BOOST_SERIALIZATION_NVP(studied);
190             ar &BOOST_SERIALIZATION_NVP(MaxGenUsedByGA);
191             ar &BOOST_SERIALIZATION_NVP(PopulationSizeUsedByGA);
192             ar &BOOST_SERIALIZATION_NVP(MutationrateUsedByGA);
193             ar &BOOST_SERIALIZATION_NVP(ElitismeUsedByGA);
194             ar &BOOST_SERIALIZATION_NVP(EndErrorUsedByGA);
195             ar &BOOST_SERIALIZATION_NVP(MaxWeightUsedByGA);
196             ar &BOOST_SERIALIZATION_NVP(MinWeightUsedByGA);
197         }
198     }
199 }
200 }
201 BOOST_CLASS_VERSION(SoilMath::NN, 0)

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8  */
9
10
11 #include "NN.h"
12 using namespace std;
13
14 namespace SoilMath {
15 NN::NN() { beta = 0.666; }
16
17 NN::NN(uint32_t inputneurons, uint32_t hiddenneurons,
18         uint32_t outputneurons) {
19     // Set the number of neurons in the network
20     inputNeurons = inputneurons;
21     hiddenNeurons = hiddenneurons;
22     outputNeurons = outputneurons;
23     // Reserve the vector space
24     iNeurons.reserve(inputNeurons + 1); // input neurons +
25     bias
26     hNeurons.reserve(hiddenNeurons + 1); // hidden neurons +
27     bias
28     oNeurons.reserve(outputNeurons); // output neurons
29
30     beta = 0.666;
31 }

```



```

74     Predict_t retVal;
75     uint32_t wCount = 0;
76
77     // Init the network
78     for (uint32_t i = 0; i < inputNeurons; i++) {
79         iNeurons.push_back(static_cast<float>(abs(input[i])));
80     }
81     for (uint32_t i = 0; i < hiddenNeurons; i++) {
82         hNeurons.push_back(0.0f);
83     }
84     for (uint32_t i = 0; i < outputNeurons; i++) {
85         oNeurons.push_back(0.0f);
86     }
87
88     for (uint32_t i = 1; i < hNeurons.size(); i++) {
89         wCount = i - 1;
90         for (uint32_t j = 0; j < iNeurons.size(); j++) {
91             hNeurons[i] += iNeurons[j] * iWeights[wCount];
92             wCount += hNeurons.size() - 1;
93         }
94         hNeurons[i] = 1 / (1 + pow(2.71828f, (-hNeurons[i] *
95             beta)));
96     }
97
98     for (uint32_t i = 0; i < oNeurons.size(); i++) {
99         wCount = i;
100        for (uint32_t j = 0; j < hNeurons.size(); j++) {
101            oNeurons[i] += hNeurons[j] * hWeights[wCount];
102            wCount += oNeurons.size();
103        }
104        oNeurons[i] =
105            (2 / (1.0f + pow(2.71828f, (-oNeurons[i] * beta)))
106            -
107            1; // Shift plus scale so the learning function can
108            be calculated
109    }
110
111    retVal.OutputNeurons = oNeurons;
112    retVal.ManualSet = false;
113    return retVal;
114 }
115
116 void NN::Learn(InputLearnVector_t input, OutputLearnVector_t
117                 cat,
118                 uint32_t noOfDescriptorsUsed __attribute__((
119                 unused))) {
120     if (optim == nullptr) {
121         optim = new SoilMath::GA(PredictLearn, inputNeurons,
122             hiddenNeurons, outputNeurons);
123     }
124     connect(optim, SIGNAL(learnErrorUpdate(double)), this,
125             SIGNAL(learnErrorUpdate(double)));
126
127     optim->Elitisme = ElitismeUsedByGA;
128     optim->EndError = EndErrorUsedByGA;
129     optim->MutationRate = MutationrateUsedByGA;

```

```
123
124     ComplexVect_t inputTest;
125     std::vector<Weight_t> weights;
126     Weight_t weight(((inputNeurons + 1) * hiddenNeurons) +
127                         ((hiddenNeurons + 1) * outputNeurons),
128                         0);
129     // loop through each case and adjust the weights
130     optim->Evolve(input, weight,
131                     MinMaxWeight_t(MinWeightUSedByGa,
132                                     MaxWeightUsedByGA), cat,
133                                     MaxGenUsedByGA, PopulationSizeUsedByGA);
134
135     this->iWeights = Weight_t(
136         weight.begin(), weight.begin() + ((inputNeurons + 1) *
137             hiddenNeurons));
138     this->hWeights = Weight_t(
139         weight.begin() + ((inputNeurons + 1) * hiddenNeurons),
140         weight.end());
141     studied = true;
142 }
143
144 void NN::SetInputNeurons(uint32_t value) {
145     if (value != inputNeurons) {
146         inputNeurons = value;
147         iNeurons.clear();
148         iNeurons.reserve(inputNeurons + 1);
149         studied = false;
150     }
151 }
152
153 void NN::SetHiddenNeurons(uint32_t value) {
154     if (value != hiddenNeurons) {
155         hiddenNeurons = value;
156         hNeurons.clear();
157         hNeurons.reserve(hiddenNeurons + 1);
158         studied = false;
159     }
160 }
161
162 void NN::SetOutputNeurons(uint32_t value) {
163     if (value != outputNeurons) {
164         outputNeurons = value;
165         oNeurons.clear();
166         oNeurons.reserve(outputNeurons);
167         studied = false;
168     }
169 }
```

F.0.4 Statistical Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (Jelle Spijker)
5   * This software is proprietary and confidential
6   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
7   */
8
9 #pragma once
10 #define MAX_UINT8_VALUE 256
11 #define VECTOR_CALC 1
12
13 #include <stdint.h>
14 #include <utility>
15 #include <vector>
16 #include <cstdlib>
17 #include <cmath>
18 #include <limits>
19 #include <typeinfo>
20 #include <string>
21
22 #include <fstream>
23
24 #include <boost/archive/binary_iarchive.hpp>
25 #include <boost/archive/binary_oarchive.hpp>
26 #include <boost/serialization/version.hpp>
27 #include <boost/math/distributions/students_t.hpp>
28
29 #include "MathException.h"
30 #include "SoilMathTypes.h"
31 #include "CommonOperations.h"
32
33 namespace SoilMath {
34
35 /*!
36  * \brief Stats class
37  * \details Usage Stats<type1, type2, type3>Stats() type 1, 2 and 3 shoudl be of
38  * the same value and concecuative in size
39 */
40
41 template <typename T1, typename T2, typename T3> class Stats
42 {
43 public:
44     bool isDiscrete = true; /*< indicates if the data is
45                             discrete or real*/
46
47     T1 *Data = nullptr;      /*< Pointer the data*/
48     uint32_t *bins = nullptr; /*< the histogram*/
49     double *CFD = nullptr;   /*< the CFD*/
50     bool Calculated = false; /*< indication if the data has
51                             been calculated*/
52     float Mean = 0.0;       /*< the mean value of the data
53                             */
54
55     /*< Constructors and Destructors
56      * \brief Default constructor
57      */
58
59     /*< Methods
60      * \brief Calculate the CDF
61      */
62
63     /*< Properties
64      * \brief Get the Data
65      */
66
67     /*< Properties
68      * \brief Get the Histogram
69      */
70
71     /*< Properties
72      * \brief Get the Mean
73      */
74
75     /*< Properties
76      * \brief Get the CDF
77      */
78
79     /*< Properties
80      * \brief Get the Calculated status
81      */
82
83     /*< Properties
84      * \brief Set the Data
85      */
86
87     /*< Properties
88      * \brief Set the Histogram
89      */
90
91     /*< Properties
92      * \brief Set the CDF
93      */
94
95     /*< Properties
96      * \brief Set the Calculated status
97      */
98
99     /*< Properties
100    * \brief Get the Data
101    */
102
103    /*< Properties
104    * \brief Get the Histogram
105    */
106
107    /*< Properties
108    * \brief Get the CDF
109    */
110
111    /*< Properties
112    * \brief Get the Calculated status
113    */
114
115    /*< Properties
116    * \brief Set the Data
117    */
118
119    /*< Properties
120    * \brief Set the Histogram
121    */
122
123    /*< Properties
124    * \brief Set the CDF
125    */
126
127    /*< Properties
128    * \brief Set the Calculated status
129    */
130
131    /*< Properties
132    * \brief Get the Data
133    */
134
135    /*< Properties
136    * \brief Get the Histogram
137    */
138
139    /*< Properties
140    * \brief Get the CDF
141    */
142
143    /*< Properties
144    * \brief Get the Calculated status
145    */
146
147    /*< Properties
148    * \brief Set the Data
149    */
150
151    /*< Properties
152    * \brief Set the Histogram
153    */
154
155    /*< Properties
156    * \brief Set the CDF
157    */
158
159    /*< Properties
160    * \brief Set the Calculated status
161    */
162
163    /*< Properties
164    * \brief Get the Data
165    */
166
167    /*< Properties
168    * \brief Get the Histogram
169    */
170
171    /*< Properties
172    * \brief Get the CDF
173    */
174
175    /*< Properties
176    * \brief Get the Calculated status
177    */
178
179    /*< Properties
180    * \brief Set the Data
181    */
182
183    /*< Properties
184    * \brief Set the Histogram
185    */
186
187    /*< Properties
188    * \brief Set the CDF
189    */
190
191    /*< Properties
192    * \brief Set the Calculated status
193    */
194
195    /*< Properties
196    * \brief Get the Data
197    */
198
199    /*< Properties
200    * \brief Get the Histogram
201    */
202
203    /*< Properties
204    * \brief Get the CDF
205    */
206
207    /*< Properties
208    * \brief Get the Calculated status
209    */
210
211    /*< Properties
212    * \brief Set the Data
213    */
214
215    /*< Properties
216    * \brief Set the Histogram
217    */
218
219    /*< Properties
220    * \brief Set the CDF
221    */
222
223    /*< Properties
224    * \brief Set the Calculated status
225    */
226
227    /*< Properties
228    * \brief Get the Data
229    */
230
231    /*< Properties
232    * \brief Get the Histogram
233    */
234
235    /*< Properties
236    * \brief Get the CDF
237    */
238
239    /*< Properties
240    * \brief Get the Calculated status
241    */
242
243    /*< Properties
244    * \brief Set the Data
245    */
246
247    /*< Properties
248    * \brief Set the Histogram
249    */
250
251    /*< Properties
252    * \brief Set the CDF
253    */
254
255    /*< Properties
256    * \brief Set the Calculated status
257    */
258
259    /*< Properties
260    * \brief Get the Data
261    */
262
263    /*< Properties
264    * \brief Get the Histogram
265    */
266
267    /*< Properties
268    * \brief Get the CDF
269    */
270
271    /*< Properties
272    * \brief Get the Calculated status
273    */
274
275    /*< Properties
276    * \brief Set the Data
277    */
278
279    /*< Properties
280    * \brief Set the Histogram
281    */
282
283    /*< Properties
284    * \brief Set the CDF
285    */
286
287    /*< Properties
288    * \brief Set the Calculated status
289    */
290
291    /*< Properties
292    * \brief Get the Data
293    */
294
295    /*< Properties
296    * \brief Get the Histogram
297    */
298
299    /*< Properties
300    * \brief Get the CDF
301    */
302
303    /*< Properties
304    * \brief Get the Calculated status
305    */
306
307    /*< Properties
308    * \brief Set the Data
309    */
310
311    /*< Properties
312    * \brief Set the Histogram
313    */
314
315    /*< Properties
316    * \brief Set the CDF
317    */
318
319    /*< Properties
320    * \brief Set the Calculated status
321    */
322
323    /*< Properties
324    * \brief Get the Data
325    */
326
327    /*< Properties
328    * \brief Get the Histogram
329    */
330
331    /*< Properties
332    * \brief Get the CDF
333    */
334
335    /*< Properties
336    * \brief Get the Calculated status
337    */
338
339    /*< Properties
340    * \brief Set the Data
341    */
342
343    /*< Properties
344    * \brief Set the Histogram
345    */
346
347    /*< Properties
348    * \brief Set the CDF
349    */
350
351    /*< Properties
352    * \brief Set the Calculated status
353    */
354
355    /*< Properties
356    * \brief Get the Data
357    */
358
359    /*< Properties
360    * \brief Get the Histogram
361    */
362
363    /*< Properties
364    * \brief Get the CDF
365    */
366
367    /*< Properties
368    * \brief Get the Calculated status
369    */
370
371    /*< Properties
372    * \brief Set the Data
373    */
374
375    /*< Properties
376    * \brief Set the Histogram
377    */
378
379    /*< Properties
380    * \brief Set the CDF
381    */
382
383    /*< Properties
384    * \brief Set the Calculated status
385    */
386
387    /*< Properties
388    * \brief Get the Data
389    */
390
391    /*< Properties
392    * \brief Get the Histogram
393    */
394
395    /*< Properties
396    * \brief Get the CDF
397    */
398
399    /*< Properties
400    * \brief Get the Calculated status
401    */
402
403    /*< Properties
404    * \brief Set the Data
405    */
406
407    /*< Properties
408    * \brief Set the Histogram
409    */
410
411    /*< Properties
412    * \brief Set the CDF
413    */
414
415    /*< Properties
416    * \brief Set the Calculated status
417    */
418
419    /*< Properties
420    * \brief Get the Data
421    */
422
423    /*< Properties
424    * \brief Get the Histogram
425    */
426
427    /*< Properties
428    * \brief Get the CDF
429    */
430
431    /*< Properties
432    * \brief Get the Calculated status
433    */
434
435    /*< Properties
436    * \brief Set the Data
437    */
438
439    /*< Properties
440    * \brief Set the Histogram
441    */
442
443    /*< Properties
444    * \brief Set the CDF
445    */
446
447    /*< Properties
448    * \brief Set the Calculated status
449    */
450
451    /*< Properties
452    * \brief Get the Data
453    */
454
455    /*< Properties
456    * \brief Get the Histogram
457    */
458
459    /*< Properties
460    * \brief Get the CDF
461    */
462
463    /*< Properties
464    * \brief Get the Calculated status
465    */
466
467    /*< Properties
468    * \brief Set the Data
469    */
470
471    /*< Properties
472    * \brief Set the Histogram
473    */
474
475    /*< Properties
476    * \brief Set the CDF
477    */
478
479    /*< Properties
480    * \brief Set the Calculated status
481    */
482
483    /*< Properties
484    * \brief Get the Data
485    */
486
487    /*< Properties
488    * \brief Get the Histogram
489    */
490
491    /*< Properties
492    * \brief Get the CDF
493    */
494
495    /*< Properties
496    * \brief Get the Calculated status
497    */
498
499    /*< Properties
500    * \brief Set the Data
501    */
502
503    /*< Properties
504    * \brief Set the Histogram
505    */
506
507    /*< Properties
508    * \brief Set the CDF
509    */
510
511    /*< Properties
512    * \brief Set the Calculated status
513    */
514
515    /*< Properties
516    * \brief Get the Data
517    */
518
519    /*< Properties
520    * \brief Get the Histogram
521    */
522
523    /*< Properties
524    * \brief Get the CDF
525    */
526
527    /*< Properties
528    * \brief Get the Calculated status
529    */
530
531    /*< Properties
532    * \brief Set the Data
533    */
534
535    /*< Properties
536    * \brief Set the Histogram
537    */
538
539    /*< Properties
540    * \brief Set the CDF
541    */
542
543    /*< Properties
544    * \brief Set the Calculated status
545    */
546
547    /*< Properties
548    * \brief Get the Data
549    */
550
551    /*< Properties
552    * \brief Get the Histogram
553    */
554
555    /*< Properties
556    * \brief Get the CDF
557    */
558
559    /*< Properties
560    * \brief Get the Calculated status
561    */
562
563    /*< Properties
564    * \brief Set the Data
565    */
566
567    /*< Properties
568    * \brief Set the Histogram
569    */
570
571    /*< Properties
572    * \brief Set the CDF
573    */
574
575    /*< Properties
576    * \brief Set the Calculated status
577    */
578
579    /*< Properties
580    * \brief Get the Data
581    */
582
583    /*< Properties
584    * \brief Get the Histogram
585    */
586
587    /*< Properties
588    * \brief Get the CDF
589    */
590
591    /*< Properties
592    * \brief Get the Calculated status
593    */
594
595    /*< Properties
596    * \brief Set the Data
597    */
598
599    /*< Properties
600    * \brief Set the Histogram
601    */
602
603    /*< Properties
604    * \brief Set the CDF
605    */
606
607    /*< Properties
608    * \brief Set the Calculated status
609    */
610
611    /*< Properties
612    * \brief Get the Data
613    */
614
615    /*< Properties
616    * \brief Get the Histogram
617    */
618
619    /*< Properties
620    * \brief Get the CDF
621    */
622
623    /*< Properties
624    * \brief Get the Calculated status
625    */
626
627    /*< Properties
628    * \brief Set the Data
629    */
630
631    /*< Properties
632    * \brief Set the Histogram
633    */
634
635    /*< Properties
636    * \brief Set the CDF
637    */
638
639    /*< Properties
640    * \brief Set the Calculated status
641    */
642
643    /*< Properties
644    * \brief Get the Data
645    */
646
647    /*< Properties
648    * \brief Get the Histogram
649    */
650
651    /*< Properties
652    * \brief Get the CDF
653    */
654
655    /*< Properties
656    * \brief Get the Calculated status
657    */
658
659    /*< Properties
660    * \brief Set the Data
661    */
662
663    /*< Properties
664    * \brief Set the Histogram
665    */
666
667    /*< Properties
668    * \brief Set the CDF
669    */
670
671    /*< Properties
672    * \brief Set the Calculated status
673    */
674
675    /*< Properties
676    * \brief Get the Data
677    */
678
679    /*< Properties
680    * \brief Get the Histogram
681    */
682
683    /*< Properties
684    * \brief Get the CDF
685    */
686
687    /*< Properties
688    * \brief Get the Calculated status
689    */
690
691    /*< Properties
692    * \brief Set the Data
693    */
694
695    /*< Properties
696    * \brief Set the Histogram
697    */
698
699    /*< Properties
700    * \brief Set the CDF
701    */
702
703    /*< Properties
704    * \brief Set the Calculated status
705    */
706
707    /*< Properties
708    * \brief Get the Data
709    */
710
711    /*< Properties
712    * \brief Get the Histogram
713    */
714
715    /*< Properties
716    * \brief Get the CDF
717    */
718
719    /*< Properties
720    * \brief Get the Calculated status
721    */
722
723    /*< Properties
724    * \brief Set the Data
725    */
726
727    /*< Properties
728    * \brief Set the Histogram
729    */
730
731    /*< Properties
732    * \brief Set the CDF
733    */
734
735    /*< Properties
736    * \brief Set the Calculated status
737    */
738
739    /*< Properties
740    * \brief Get the Data
741    */
742
743    /*< Properties
744    * \brief Get the Histogram
745    */
746
747    /*< Properties
748    * \brief Get the CDF
749    */
750
751    /*< Properties
752    * \brief Get the Calculated status
753    */
754
755    /*< Properties
756    * \brief Set the Data
757    */
758
759    /*< Properties
760    * \brief Set the Histogram
761    */
762
763    /*< Properties
764    * \brief Set the CDF
765    */
766
767    /*< Properties
768    * \brief Set the Calculated status
769    */
770
771    /*< Properties
772    * \brief Get the Data
773    */
774
775    /*< Properties
776    * \brief Get the Histogram
777    */
778
779    /*< Properties
780    * \brief Get the CDF
781    */
782
783    /*< Properties
784    * \brief Get the Calculated status
785    */
786
787    /*< Properties
788    * \brief Set the Data
789    */
790
791    /*< Properties
792    * \brief Set the Histogram
793    */
794
795    /*< Properties
796    * \brief Set the CDF
797    */
798
799    /*< Properties
800    * \brief Set the Calculated status
801    */
802
803    /*< Properties
804    * \brief Get the Data
805    */
806
807    /*< Properties
808    * \brief Get the Histogram
809    */
810
811    /*< Properties
812    * \brief Get the CDF
813    */
814
815    /*< Properties
816    * \brief Get the Calculated status
817    */
818
819    /*< Properties
820    * \brief Set the Data
821    */
822
823    /*< Properties
824    * \brief Set the Histogram
825    */
826
827    /*< Properties
828    * \brief Set the CDF
829    */
830
831    /*< Properties
832    * \brief Set the Calculated status
833    */
834
835    /*< Properties
836    * \brief Get the Data
837    */
838
839    /*< Properties
840    * \brief Get the Histogram
841    */
842
843    /*< Properties
844    * \brief Get the CDF
845    */
846
847    /*< Properties
848    * \brief Get the Calculated status
849    */
850
851    /*< Properties
852    * \brief Set the Data
853    */
854
855    /*< Properties
856    * \brief Set the Histogram
857    */
858
859    /*< Properties
860    * \brief Set the CDF
861    */
862
863    /*< Properties
864    * \brief Set the Calculated status
865    */
866
867    /*< Properties
868    * \brief Get the Data
869    */
870
871    /*< Properties
872    * \brief Get the Histogram
873    */
874
875    /*< Properties
876    * \brief Get the CDF
877    */
878
879    /*< Properties
880    * \brief Get the Calculated status
881    */
882
883    /*< Properties
884    * \brief Set the Data
885    */
886
887    /*< Properties
888    * \brief Set the Histogram
889    */
890
891    /*< Properties
892    * \brief Set the CDF
893    */
894
895    /*< Properties
896    * \brief Set the Calculated status
897    */
898
899    /*< Properties
900    * \brief Get the Data
901    */
902
903    /*< Properties
904    * \brief Get the Histogram
905    */
906
907    /*< Properties
908    * \brief Get the CDF
909    */
910
911    /*< Properties
912    * \brief Get the Calculated status
913    */
914
915    /*< Properties
916    * \brief Set the Data
917    */
918
919    /*< Properties
920    * \brief Set the Histogram
921    */
922
923    /*< Properties
924    * \brief Set the CDF
925    */
926
927    /*< Properties
928    * \brief Set the Calculated status
929    */
930
931    /*< Properties
932    * \brief Get the Data
933    */
934
935    /*< Properties
936    * \brief Get the Histogram
937    */
938
939    /*< Properties
940    * \brief Get the CDF
941    */
942
943    /*< Properties
944    * \brief Get the Calculated status
945    */
946
947    /*< Properties
948    * \brief Set the Data
949    */
950
951    /*< Properties
952    * \brief Set the Histogram
953    */
954
955    /*< Properties
956    * \brief Set the CDF
957    */
958
959    /*< Properties
960    * \brief Set the Calculated status
961    */
962
963    /*< Properties
964    * \brief Get the Data
965    */
966
967    /*< Properties
968    * \brief Get the Histogram
969    */
970
971    /*< Properties
972    * \brief Get the CDF
973    */
974
975    /*< Properties
976    * \brief Get the Calculated status
977    */
978
979    /*< Properties
980    * \brief Set the Data
981    */
982
983    /*< Properties
984    * \brief Set the Histogram
985    */
986
987    /*< Properties
988    * \brief Set the CDF
989    */
990
991    /*< Properties
992    * \brief Set the Calculated status
993    */
994
995    /*< Properties
996    * \brief Get the Data
997    */
998
999    /*< Properties
1000   * \brief Get the Histogram
1001   */
1002
1003    /*< Properties
1004   * \brief Get the CDF
1005   */
1006
1007    /*< Properties
1008   * \brief Get the Calculated status
1009   */
1010
1011    /*< Properties
1012   * \brief Set the Data
1013   */
1014
1015    /*< Properties
1016   * \brief Set the Histogram
1017   */
1018
1019    /*< Properties
1020   * \brief Set the CDF
1021   */
1022
1023    /*< Properties
1024   * \brief Set the Calculated status
1025   */
1026
1027    /*< Properties
1028   * \brief Get the Data
1029   */
1030
1031    /*< Properties
1032   * \brief Get the Histogram
1033   */
1034
1035    /*< Properties
1036   * \brief Get the CDF
1037   */
1038
1039    /*< Properties
1040   * \brief Get the Calculated status
1041   */
1042
1043    /*< Properties
1044   * \brief Set the Data
1045   */
1046
1047    /*< Properties
1048   * \brief Set the Histogram
1049   */
1050
1051    /*< Properties
1052   * \brief Set the CDF
1053   */
1054
1055    /*< Properties
1056   * \brief Set the Calculated status
1057   */
1058
1059    /*< Properties
1060   * \brief Get the Data
1061   */
1062
1063    /*< Properties
1064   * \brief Get the Histogram
1065   */
1066
1067    /*< Properties
1068   * \brief Get the CDF
1069   */
1070
1071    /*< Properties
1072   * \brief Get the Calculated status
1073   */
1074
1075    /*< Properties
1076   * \brief Set the Data
1077   */
1078
1079    /*< Properties
1080   * \brief Set the Histogram
1081   */
1082
1083    /*< Properties
1084   * \brief Set the CDF
1085   */
1086
1087    /*< Properties
1088   * \brief Set the Calculated status
1089   */
1090
1091    /*< Properties
1092   * \brief Get the Data
1093   */
1094
1095    /*< Properties
1096   * \brief Get the Histogram
1097   */
1098
1099    /*< Properties
1100   * \brief Get the CDF
1101   */
1102
1103    /*< Properties
1104   * \brief Get the Calculated status
1105   */
1106
1107    /*< Properties
1108   * \brief Set the Data
1109   */
1110
1111    /*< Properties
1112   * \brief Set the Histogram
1113   */
1114
1115    /*< Properties
1116   * \brief Set the CDF
1117   */
1118
1119    /*< Properties
1120   * \brief Set the Calculated status
1121   */
1122
1123    /*< Properties
1124   * \brief Get the Data
1125   */
1126
1127    /*< Properties
1128   * \brief Get the Histogram
1129   */
1130
1131    /*< Properties
1132   * \brief Get the CDF
1133   */
1134
1135    /*< Properties
1136   * \brief Get the Calculated status
1137   */
1138
1139    /*< Properties
1140   * \brief Set the Data
1141   */
1142
1143    /*< Properties
1144   * \brief Set the Histogram
1145   */
1146
1147    /*< Properties
1148   * \brief Set the CDF
1149   */
1150
1151    /*< Properties
1152   * \brief Set the Calculated status
1153   */
1154
1155    /*< Properties
1156   * \brief Get the Data
1157   */
1158
1159    /*< Properties
1160   * \brief Get the Histogram
1161   */
1162
1163    /*< Properties
1164   * \brief Get the CDF
1165   */
1166
1167    /*< Properties
1168   * \brief Get the Calculated status
1169   */
1170
1171    /*< Properties
1172   * \brief Set the Data
1173   */
1174
1175    /*< Properties
1176   * \brief Set the Histogram
1177   */
1178
1179    /*< Properties
1180   * \brief Set the CDF
1181   */
1182
1183    /*< Properties
1184   * \brief Set the Calculated status
1185   */
1186
1187    /*< Properties
1188   * \brief Get the Data
1189   */
1190
1191    /*< Properties
1192   * \brief Get the Histogram
1193   */
1194
1195    /*< Properties
1196   * \brief Get the CDF
1197   */
1198
1199    /*< Properties
1200   * \brief Get the Calculated status
1201   */
1202
1203    /*< Properties
1204   * \brief Set the Data
1205   */
1206
1207    /*< Properties
1208   * \brief Set the Histogram
1209   */
1210
1211    /*< Properties
1212   * \brief Set the CDF
1213   */
1214
1215    /*< Properties
1216   * \brief Set the Calculated status
1217   */
1218
1219    /*< Properties
1220   * \brief Get the Data
1221   */
1222
1223    /*< Properties
1224   * \brief Get the Histogram
1225   */
1226
1227    /*< Properties
1228   * \brief Get the CDF
1229   */
1230
1231    /*< Properties
1232   * \brief Get the Calculated status
1233   */
1234
1235    /*< Properties
1236   * \brief Set the Data
1237   */
1238
1239    /*< Properties
1240   * \brief Set the Histogram
1241   */
1242
1243    /*< Properties
1244   * \brief Set the CDF
1245   */
1246
1247    /*< Properties
1248   * \brief Set the Calculated status
1249   */
1250
1251    /*< Properties
1252   * \brief Get the Data
1253   */
1254
1255    /*< Properties
1256   * \brief Get the Histogram
1257   */
1258
1259    /*< Properties
1260   * \brief Get the CDF
1261   */
1262
1263    /*< Properties
1264   * \brief Get the Calculated status
1265   */
1266
1267    /*< Properties
1268   * \brief Set the Data
1269   */
1270
1271    /*< Properties
1272   * \brief Set the Histogram
1273   */
1274
1275    /*< Properties
1276   * \brief Set the CDF
1277   */
1278
1279    /*< Properties
1280   * \brief Set the Calculated status
1281   */
1282
1283    /*< Properties
1284   * \brief Get the Data
1285   */
1286
1287    /*< Properties
1288   * \brief Get the Histogram
1289   */
1290
1291    /*< Properties
1292   * \brief Get the CDF
1293   */
1294
1295    /*< Properties
1296   * \brief Get the Calculated status
1297   */
1298
1299    /*< Properties
1300   * \brief Set the Data
1301   */
1302
1303    /*< Properties
1304   * \brief Set the Histogram
1305   */
1306
1307    /*< Properties
1308   * \brief Set the CDF
1309   */
1310
1311    /*< Properties
1312   * \brief Set the Calculated status
1313   */
1314
1315    /*< Properties
1316   * \brief Get the Data
1317   */
1318
1319    /*< Properties
1320   * \brief Get the Histogram
1321   */
1322
1323    /*< Properties
1324   * \brief Get the CDF
1325   */
1326
1327    /*< Properties
1328   * \brief Get the Calculated status
1329   */
1330
1331    /*< Properties
1332   * \brief Set the Data
1333   */
1334
1335    /*< Properties
1336   * \brief Set the Histogram
1337   */
1338
1339    /*< Properties
1340   * \brief Set the CDF
1341   */
1342
1343    /*< Properties
1344   * \brief Set the Calculated status
1345   */
1346
1347    /*< Properties
1348   * \brief Get the Data
1349   */
1350
1351    /*< Properties
1352   * \brief Get the Histogram
1353   */
1354
1355    /*< Properties
1356   * \brief Get the CDF
1357   */
1358
1359    /*< Properties
1360   * \brief Get the Calculated status
1361   */
1362
1363    /*< Properties
1364   * \brief Set the Data
1365   */
1366
1367    /*< Properties
1368   * \brief Set the Histogram
1369   */
1370
1371    /*< Properties
1372   * \brief Set the CDF
1373   */
1374
1375    /*< Properties
1376   * \brief Set the Calculated status
1377   */
1378
1379    /*< Properties
1380   * \brief Get the Data
1381   */
1382
1383    /*< Properties
1384   * \brief Get the Histogram
1385   */
1386
1387    /*< Properties
1388   * \brief Get the CDF
1389   */
1390
1391    /*< Properties
1392   * \brief Get the Calculated status
1393   */
1394
1395    /*< Properties
1396   * \brief Set the Data
1397   */
1398
1399    /*< Properties
1400   * \brief Set the Histogram
1401   */
1402
1403    /*< Properties
1404   * \brief Set the CDF
1405   */
1406
1407    /*< Properties
1408   * \brief Set the Calculated status
1409   */
1410
1411    /*< Properties
1412   * \brief Get the Data
1413   */
1414
1415    /*< Properties
1416   * \brief Get the Histogram
1417   */
1418
1419    /*< Properties
1420   * \brief Get the CDF
1421   */
1422
1423    /*< Properties
1424   * \brief Get the Calculated status
1425   */
1426
1427    /*< Properties
1428   * \brief Set the Data
1429   */
1430
1431    /*< Properties
1432   * \brief Set the Histogram
1433   */
1434
1435    /*< Properties
1436   * \brief Set the CDF
1437   */
1438
1439    /*< Properties
1440   * \brief Set the Calculated status
1441   */
1442
1443    /*< Properties
1444   * \brief Get the Data
1445   */
1446
1447    /*< Properties
1448   * \brief Get the Histogram
1449   */
1450
1451    /*< Properties
1452   * \brief Get the CDF
1453   */
1454
1455    /*< Properties
1456   * \brief Get the Calculated status
1457   */
1458
1459    /*< Properties
1460   * \brief Set the Data
1461   */
1462
1463    /*< Properties
1464   * \brief Set the Histogram
1465   */
1466
1467    /*< Properties
1468   * \brief Set the CDF
1469   */
1470
1471    /*< Properties
1472   * \brief Set the Calculated status
1473   */
1474
1475    /*< Properties
1476   * \brief Get the Data
1477   */
1478
1479    /*< Properties
1480   * \brief Get the Histogram
1481   */
1482
1483    /*< Properties
1484   * \brief Get the CDF
1485   */
1486
1487    /*< Properties
1488   * \brief Get the Calculated status
1489   */
1490
1491    /*< Properties
1492   * \brief Set the Data
1493   */
1494
1495    /*< Properties
1496   * \brief Set the Histogram
1497   */
1498
1499    /*< Properties
1500   * \brief Set the CDF
1501   */
1502
1503    /*< Properties
1504   * \brief Set the Calculated status
1505   */
1506
1507    /*< Properties
1508   * \brief Get the Data
1509   */
1510
1511    /*< Properties
1512   * \brief Get the Histogram
1513   */
1514
1515    /*< Properties
1516   * \brief Get the CDF
1517   */
1518
1519    /*< Properties
1520   * \brief Get the Calculated status
1521   */
1522
1523    /*< Properties
1524
```

```

48     uint32_t n = 0;           /**< number of data points*/
49     uint32_t noBins = 0;     /**< number of bins*/
50     T1 Range = 0;           /**< range of the data*/
51     T1 min = 0;             /**< minimum value*/
52     T1 max = 0;             /**< maximum value*/
53     T1 Startbin = 0;        /**< First bin value*/
54     T1 EndBin = 0;           /**< End bin value*/
55     T1 binRange = 0;         /**< the range of a single bin*/
56     float Std = 0.0;         /**< standard deviation*/
57     T3 Sum = 0;             /**< total sum of all the data
58     values*/
58     uint16_t Rows = 0;        /**< number of rows from the
59     data matrix*/
59     uint16_t Cols = 0;        /**< number of cols from the
60     data matrix*/
60     bool StartAtZero = true;  /**< indication of the minimum
61     value starts at zero
61                         or could be less*/
62     double *BinRanges = nullptr;
63     double HighestPDF = 0.;
64
65     uint32_t *begin() { return &bins[0]; }    /**< pointer to
66     the first bin*/
66     uint32_t *end() { return &bins[noBins]; } /**< pointer to
67     the last + 1 bin*/
67
68 /*!
69  * \brief WelchTest Compare the sample using the Welch's
70  * Test
71  * \details (source:
72  *   * http://www.boost.org/doc/libs/1\_57\_0/libs/math/doc/html/math\_toolkit/stat\_tut/weg/st\_eg/two\_sample\_students\_t.html)
73  * \param statComp Statiscs Results of which it should be
74  * tested against
75  * \return
76 */
77
78     bool WelchTest(SoilMath::Stats<T1, T2, T3> &statComp) {
79     double alpha = 0.05;
80     // Degrees of freedom:
81     double v = statComp.Std * statComp.Std / statComp.n +
82     this->Std * this->Std / this->n;
83     v *= v;
84     double t1 = statComp.Std * statComp.Std / statComp.n;
85     t1 *= t1;
86     t1 /= (statComp.n - 1);
87     double t2 = this->Std * this->Std / this->n;
88     t2 *= t2;
89     t2 /= (this->n - 1);
90     v /= (t1 + t2);
91     // t-statistic:
92     double t_stat = (statComp.Mean - this->Mean) /
93     sqrt(statComp.Std * statComp.Std /
94           statComp.n +
95           this->Std * this->Std / this->n);
96
97

```

```

93     // Define our distribution, and get the probability:
94     //
95     boost::math::students_t dist(v);
96     double q = cdf(complement(dist, fabs(t_stat)));
97
98     bool rejected = false;
99     // Sample 1 Mean == Sample 2 Mean test the NULL
100    // hypothesis, the two means
101    // are the same
102    if (q < alpha / 2)
103        rejected = false;
104    else
105        rejected = true;
106    return rejected;
107 }
108 */
109 * \brief Stats Constructor
110 * \param rhs Right hand side
111 */
112 Stats(const Stats &rhs)
113     : bins{new uint32_t[rhs.noBins]{0}}, CFD{new double[
114         rhs.noBins]{}}, 
115         BinRanges{new double[rhs.noBins]{}} {
116     this->binRange = rhs.binRange;
117     this->Calculated = rhs.Calculated;
118     this->Cols = rhs.Cols;
119     this->EndBin = rhs.EndBin;
120     this->isDiscrete = rhs.isDiscrete;
121     this->max = rhs.max;
122     this->Mean = rhs.Mean;
123     this->min = rhs.min;
124     this->n = rhs.n;
125     this->noBins = rhs.noBins;
126     this->n_end = rhs.n_end;
127     this->Range = rhs.Range;
128     this->Rows = rhs.Rows;
129     this->Startbin = rhs.Startbin;
130     this->Std = rhs.Std;
131     this->Sum = rhs.Sum;
132     std::copy(rhs.bins, rhs.bins + rhs.noBins, this->bins);
133     std::copy(rhs.CFD, rhs.CFD + rhs.noBins, this->CFD);
134     std::copy(rhs.BinRanges, rhs.BinRanges + rhs.noBins,
135             this->BinRanges);
136     this->Data = rhs.Data;
137     this->StartAtZero = rhs.StartAtZero;
138     this->HighestPDF = rhs.HighestPDF;
139 }
140 */
141 * \brief operator = Assigment operator
142 * \param rhs right hand side
143 * \return returns the right hand side
144 */
145 Stats &operator=(Stats const &rhs) {
146     if (&rhs != this) {

```

```

146     Data = rhs.Data;
147
148     if (bins != nullptr) {
149         delete[] bins;
150         bins = nullptr;
151     }
152     if (CFD != nullptr) {
153         delete[] CFD;
154         CFD = nullptr;
155     }
156     if (BinRanges != nullptr) {
157         delete[] BinRanges;
158         BinRanges = nullptr;
159     }
160
161     bins = new uint32_t[rhs.noBins];      // leak
162     CFD = new double[rhs.noBins];        // leak
163     BinRanges = new double[rhs.noBins];   // leak
164     this->binRange = rhs.binRange;
165     this->Calculated = rhs.Calculated;
166     this->Cols = rhs.Cols;
167     this->EndBin = rhs.EndBin;
168     this->isDiscrete = rhs.isDiscrete;
169     this->max = rhs.max;
170     this->Mean = rhs.Mean;
171     this->min = rhs.min;
172     this->n = rhs.n;
173     this->noBins = rhs.noBins;
174     this->n_end = rhs.n_end;
175     this->Range = rhs.Range;
176     this->Rows = rhs.Rows;
177     this->Startbin = rhs.Startbin;
178     this->Std = rhs.Std;
179     this->Sum = rhs.Sum;
180     this->Data = &rhs.Data[0];
181     std::copy(rhs.bins, rhs.bins + rhs.noBins, this->bins)
182         ;
183     std::copy(rhs.CFD, rhs.CFD + rhs.noBins, this->CFD);
184     std::copy(rhs.BinRanges, rhs.BinRanges + rhs.noBins,
185               this->BinRanges);
186     this->StartAtZero = rhs.StartAtZero;
187     this->HighestPDF = rhs.HighestPDF;
188 }
189
190 */
191 * \brief Stats Constructor
192 * \param noBins number of bins with which to build the
193 * histogram
194 * \param startBin starting value of the first bin
195 * \param endBin end value of the second bin
196 Stats(int noBins = 256, T1 startBin = 0, T1 endBin = 255)
197 {
198     min = std::numeric_limits<T1>::max();

```

```

198     max = std::numeric_limits<T1>::min();
199     Range = std::numeric_limits<T1>::max();
200     Startbin = startBin;
201     EndBin = endBin;
202     this->noBins = noBins;
203     bins = new uint32_t[noBins]{0};      // leak
204     CFD = new double[noBins]{};        // leak
205     BinRanges = new double[noBins]{}; // leak
206
207     if (typeid(T1) == typeid(float) || typeid(T1) == typeid(
208         double) ||
209         typeid(T1) == typeid(long double)) {
210         isDiscrete = false;
211         binRange = static_cast<T1>((EndBin - Startbin) /
212             noBins);
213     } else {
214         isDiscrete = true;
215         binRange = static_cast<T1>(round((EndBin - Startbin) /
216             noBins));
217     }
218
219     /*!
220      * \brief Stats constructor
221      * \param data Pointer to the data
222      * \param rows Number of rows
223      * \param cols Number of Columns
224      * \param noBins Number of bins
225      * \param startBin Value of the start bin
226      * \param startatzero bool indicating if the bins should
227      *   be shifted from zero
228      */
229     Stats(T1 *data, uint16_t rows, uint16_t cols, int noBins =
230           256,
231           T1 startBin = 0, bool startatzero = true) {
232     min = std::numeric_limits<T1>::max();
233     max = std::numeric_limits<T1>::min();
234     Range = max - min;
235
236     Startbin = startBin;
237     EndBin = startBin + noBins;
238     StartAtZero = startatzero;
239
240     if (typeid(T1) == typeid(float) || typeid(T1) == typeid(
241         double) ||
242         typeid(T1) == typeid(long double)) {
243         isDiscrete = false;
244     } else {
245         isDiscrete = true;
246     }
247
248     Data = data;
249     Rows = rows;
250     Cols = cols;
251     bins = new uint32_t[noBins]{0};
252     CFD = new double[noBins]{};

```

```

248     BinRanges = new double[noBins]{};
249     this->noBins = noBins;
250     if (isDiscrete) {
251         BasicCalculate();
252     } else {
253         BasicCalculateFloat();
254     }
255 }
256
257 /*!
258 * \brief Stats Constructor
259 * \param data Pointer the data
260 * \param rows Number of rows
261 * \param cols Number of Columns
262 * \param mask the mask should have the same size as the
263 *   data a value of zero
264 * indicates that the data pointer doesn't exist. A 1
265 *   indicates that the data
266 * pointer is to be used
267 * \param noBins Number of bins
268 * \param startBin Value of the start bin
269 * \param startatzero indicating if the bins should be
270 *   shifted from zero
271 */
272 Stats(T1 *data, uint16_t rows, uint16_t cols, uchar *mask,
273       int noBins = 256,
274       T1 startBin = 0, bool startatzero = true) {
275     min = std::numeric_limits<T1>::max();
276     max = std::numeric_limits<T1>::min();
277     Range = max - min;
278
279     Startbin = startBin;
280     EndBin = startBin + noBins;
281     StartAtZero = startatzero;
282
283     if (typeid(T1) == typeid(float) || typeid(T1) == typeid(
284         double) ||
285         typeid(T1) == typeid(long double)) {
286         isDiscrete = false;
287     } else {
288         isDiscrete = true;
289     }
290
291     Data = data;
292     Rows = rows;
293     Cols = cols;
294     bins = new uint32_t[noBins]{0};
295     CFD = new double[noBins]{};
296     BinRanges = new double[noBins]{};
297     this->noBins = noBins;
298     if (isDiscrete) {
299         BasicCalculate(mask);
300     } else {
301         BasicCalculateFloat(mask);
302     }
303 }

```

```

299
300  /*!
301  * \brief Stats Constructor
302  * \param binData The histogram data
303  * \param startC start counter
304  * \param endC end counter
305  */
306  Stats(T2 *binData, uint16_t startC, uint16_t endC) {
307      noBins = endC - startC;
308      Startbin = startC;
309      EndBin = endC;
310      uint32_t i = noBins;
311
312      if (typeid(T1) == typeid(float) || typeid(T1) == typeid(
313          double) ||
314          typeid(T1) == typeid(long double)) {
315          isDiscrete = false;
316          throw Exception::MathException(
317              EXCEPTION_TYPE_NOT_SUPPORTED,
318              EXCEPTION_TYPE_NOT_SUPPORTED_NR
319          );
320
321      bins = new uint32_t[noBins]{0};
322      CFD = new double[noBins]{};
323      BinRanges = new double[noBins]{};
324      while (i-- > 0) {
325          bins[i] = binData[i];
326          n += binData[i];
327      }
328      BinCalculations(startC, endC);
329  }
330
331  ~Stats() {
332      Data == nullptr;
333      if (bins != nullptr) {
334          delete[] bins;
335          bins = nullptr;
336      }
337      if (CFD != nullptr) {
338          delete[] CFD;
339          CFD = nullptr;
340      }
341      if (BinRanges != nullptr) {
342          delete[] BinRanges;
343          BinRanges = nullptr;
344      }
345  }
346
347  /*!
348  * \brief BasicCalculateFloat execute the basic float data
349  * calculations
350  */
void BasicCalculateFloat() {

```

```

351     float sum_dev = 0.0;
352     n = Rows * Cols;
353     for (uint32_t i = 0; i < n; i++) {
354         if (Data[i] > max) {
355             max = Data[i];
356         }
357         if (Data[i] < min) {
358             min = Data[i];
359         }
360         Sum += Data[i];
361     }
362     binRange = (max - min) / noBins;
363     uint32_t index = 0;
364     Mean = Sum / (float)n;
365     Range = max - min;
366
367     if (StartAtZero) {
368         for (uint32_t i = 0; i < n; i++) {
369             index = static_cast<uint32_t>(Data[i] / binRange);
370             if (index == noBins) {
371                 index -= 1;
372             }
373             bins[index]++;
374             sum_dev += pow((Data[i] - Mean), 2);
375         }
376     } else {
377         for (uint32_t i = 0; i < n; i++) {
378             index = static_cast<uint32_t>((Data[i] - min) /
379                                         binRange);
380             if (index == noBins) {
381                 index -= 1;
382             }
383             bins[index]++;
384             sum_dev += pow((Data[i] - Mean), 2);
385         }
386     }
387     Std = sqrt((float)(sum_dev / n));
388     getCFD();
389     Calculated = true;
390 }
391 /**
392 * \brief BasicCalculateFloat execute the basic float data
393 * calculations with a
394 * mask
395 * \param mask uchar mask type 0 don't calculate, 1
396 * calculate
397 */
398 void BasicCalculateFloat(uchar *mask) {
399     float sum_dev = 0.0;
400     n = Rows * Cols;
401     uint32_t nmask = 0;
402     for (uint32_t i = 0; i < n; i++) {
403         if (mask[i] != 0) {
404             if (Data[i] > max) {
405                 max = Data[i];
406             }
407             sum_dev += Data[i];
408         }
409     }
410     Mean = sum_dev / n;
411     Std = sqrt((float)(sum_dev / n));
412     getCFD();
413     Calculated = true;
414 }
```

```

404
405     if (Data[i] < min) {
406         min = Data[i];
407     }
408     Sum += Data[i];
409     nmask++;
410 }
411 }
412 binRange = (max - min) / noBins;
413 uint32_t index = 0;
414 Mean = Sum / (float)nmask;
415 Range = max - min;
416 if (StartAtZero) {
417     for (uint32_t i = 0; i < n; i++) {
418         if (mask[i] != 0) {
419             index = static_cast<uint32_t>(Data[i] / binRange);
420             if (index == noBins) {
421                 index -= 1;
422             }
423             bins[index]++;
424             sum_dev += pow((Data[i] - Mean), 2);
425         }
426     }
427 } else {
428     for (uint32_t i = 0; i < n; i++) {
429         if (mask[i] != 0) {
430             index = static_cast<uint32_t>((Data[i] - min) /
431                 binRange);
432             if (index == noBins) {
433                 index -= 1;
434             }
435             bins[index]++;
436             sum_dev += pow((Data[i] - Mean), 2);
437         }
438     }
439     Std = sqrt((float)(sum_dev / nmask));
440     getCFD();
441     Calculated = true;
442 }
443
444 /**
445 * \brief BasicCalculate execute the basic discrete data
446 *        calculations
447 */
448 void BasicCalculate() {
449     double sum_dev = 0.0;
450     n = Rows * Cols;
451     for (uint32_t i = 0; i < n; i++) {
452         if (Data[i] > max) {
453             max = Data[i];
454         }
455         if (Data[i] < min) {
456             min = Data[i];
457         }
458         Sum += Data[i];

```

```

458     }
459     binRange = static_cast<T1>(ceil((max - min) /
460         static_cast<float>(noBins)));
460     if (binRange == 0) {
461         binRange = 1;
462     }
463     Mean = Sum / (float)n;
464     Range = max - min;
465
466     uint32_t index;
467     if (StartAtZero) {
468         std::for_each(Data, Data + n, [&](T1 &d) {
469             index = static_cast<uint32_t>(d / binRange);
470             if (index == noBins) {
471                 index -= 1;
472             }
473             bins[index]++;
474             sum_dev += pow((d - Mean), 2);
475         });
476     } else {
477         std::for_each(Data, Data + n, [&](T1 &d) {
478             index = static_cast<uint32_t>((d - min) / binRange);
479             if (index == noBins) {
480                 index -= 1;
481             }
482             bins[index]++;
483             sum_dev += pow((d - Mean), 2);
484         });
485     }
486     Std = sqrt((float)(sum_dev / n));
487     getCFD();
488     Calculated = true;
489 }
490
491 /**
492 * \brief BasicCalculate execute the basic discrete data
493 * calculations with
494 * mask
495 * \param mask uchar mask type 0 don't calculate, 1
496 * calculate
497 */
498 void BasicCalculate(uchar *mask) {
499     double sum_dev = 0.0;
500     n = Rows * Cols;
501     uint32_t nmask = 0;
502     uint32_t i = 0;
503     std::for_each(Data, Data + n, [&](T1 &d) {
504         if (mask[i++] != 0) {
505             if (d > max) {
506                 max = d;
507             }
508             if (d < min) {
509                 min = d;
510             }
511             Sum += d;
512             nmask++;
513         }
514     });
515 }
```

```

511         }
512     );
513     binRange = static_cast<T1>(ceil((max - min) /
514         static_cast<float>(noBins)));
514     Mean = Sum / (float)nmask;
515     Range = max - min;
516
517     uint32_t index;
518     if (StartAtZero) {
519         i = 0;
520         std::for_each(Data, Data + n, [&](T1 &d) {
521             if (mask[i++] != 0) {
522                 index = static_cast<uint32_t>(d / binRange);
523                 if (index == noBins) {
524                     index -= 1;
525                 }
526                 bins[index]++;
527                 sum_dev += pow((d - Mean), 2);
528             }
529         });
530     } else {
531         i = 0;
532         std::for_each(Data, Data + n, [&](T1 &d) {
533             if (mask[i++] != 0) {
534                 index = static_cast<uint32_t>((d - min) / binRange
535                     );
536                 if (index == noBins) {
537                     index -= 1;
538                 }
539                 bins[index]++;
540                 sum_dev += pow((d - Mean), 2);
541             }
542         });
543         Std = sqrt((float)(sum_dev / nmask));
544         getCFD();
545         Calculated = true;
546     }
547
548 /**
549  * \brief BinCalculations execute the calculations with the
550  * histogram
551  * \param startC start counter
552  * \param endC end counter
553  */
554 void BinCalculations(uint16_t startC, uint16_t endC
555     __attribute__((unused))) {
556     float sum_dev = 0.0;
557     // Get the Sum
558     uint32_t i = 0;
559     for_each(begin(), end(), [&](uint32_t &b) { Sum += b * (
560         startC + i++); });
561
562     // Get Mean
563     Mean = Sum / (float)n;
564
565     // Get Std Deviation
566     Std = sqrt((float)(sum_dev / nmask));
567
568     // Get CDF
569     getCFD();
570
571     Calculated = true;
572 }

```

```

562     // Get max
563     for (int i = noBins - 1; i >= 0; i--) {
564         if (bins[i] != 0) {
565             max = i + startC;
566             break;
567         }
568     }
569
570     // Get min
571     for (uint32_t i = 0; i < noBins; i++) {
572         if (bins[i] != 0) {
573             min = i + startC;
574             break;
575         }
576     }
577
578     // Get Range;
579     Range = max - min;
580
581     // Calculate Standard Deviation
582     i = 0;
583     for_each(begin(), end(), [&](uint32_t &b) {
584         sum_dev += b * pow(((i++ + startC) - Mean), 2);
585     });
586     Std = sqrt((float)(sum_dev / n));
587     getCFD();
588     Calculated = true;
589 }
590
591     uint32_t HighestFrequency() {
592         uint32_t freq = 0;
593         std::for_each(begin(), end(), [&](uint32_t &B) {
594             if (B > freq) {
595                 freq = B;
596             }
597         });
598         return freq;
599     }
600
601     void GetPDFfunction(std::vector<double> &xAxis, std::
602         vector<double> &yAxis,
603             double Step, double start = 0, double
604             stop = 7) {
605
606         uint32_t resolution;
607         resolution = static_cast<uint32_t>(((stop - start) /
608             Step) + 0.5);
609
610         xAxis.push_back(start);
611         double yVal0 = (1 / (Std * 2.506628274631)) *
612             exp(-(pow((start - Mean), 2) / (2 * pow(
613                 Std, 2))));
614         yAxis.push_back(yVal0);
615         HighestPDF = yVal0;
616         for (uint32_t i = 1; i < resolution; i++) {
617             double xVal = xAxis[xAxis.size() - 1] + Step;
618             xAxis.push_back(xVal);

```

```

614     double yVal = (1 / (Std * 2.506628274631)) *
615         exp(-(pow((xVal - Mean), 2) / (2 * pow(
616             Std, 2))));;
617     yAxis.push_back(yVal);
618     if (yVal > HighestPDF) {
619         HighestPDF = yVal;
620     }
621 }
622
623 protected:
624     uint32_t n_end = 0; /*< data end counter used with mask*/
625
626 /**
627  * \brief getCFD get the CFD matrix;
628  */
629 void getCFD() {
630     uint32_t *sumBin = new uint32_t[noBins];
631     sumBin[0] = bins[0];
632     CFD[0] = (static_cast<double>(sumBin[0]) / static_cast<
633         double>(n)) * 100.;
634     for (uint32_t i = 1; i < noBins; i++) {
635         sumBin[i] = (sumBin[i - 1] + bins[i]);
636         CFD[i] = (static_cast<double>(sumBin[i]) / static_cast
637             <double>(n)) * 100.;
638         if (CFD[i] > HighestPDF) {
639             HighestPDF = CFD[i];
640         }
641     }
642     delete[] sumBin;
643 }
644
645 friend class boost::serialization::access; /*<
646     Serialization class*/
647
648 /**
649  * \brief serialize the object
650  * \param ar argument
651  * \param version
652  */
653 template <class Archive>
654 void serialize(Archive &ar, const unsigned int version) {
655     if (version == 0) {
656         ar &isDiscrete;
657         ar &n;
658         ar &noBins;
659         for (size_t dc = 0; dc < noBins; dc++) {
660             ar &bins[dc];
661         }
662         for (size_t dc = 0; dc < noBins; dc++) {
663             ar &CFD[dc];
664         }
665         for (size_t dc = 0; dc < noBins; dc++) {
666             ar &BinRanges[dc];
667         }
668         ar &Calculated;

```

```

666     ar &Mean;
667     ar &Range;
668     ar &min;
669     ar &max;
670     ar &Startbin;
671     ar &EndBin;
672     ar &binRange;
673     ar &Std;
674     ar &Sum;
675     ar &Rows;
676     ar &Cols;
677     ar &StartAtZero;
678     ar &HighestPDF;
679 }
680 }
681 };
682 }
683
684 typedef SoilMath::Stats<float, double, long double>
685     floatStat_t; /*< floating Stat type*/
686 typedef SoilMath::Stats<uchar, uint32_t, uint64_t>
687     ucharStat_t; /*< uchar Stat type*/
688 typedef SoilMath::Stats<uint16_t, uint32_t, uint64_t>
689     uint16Stat_t; /*< uint16 Stat type*/
690 typedef SoilMath::Stats<uint32_t, uint32_t, uint64_t>
691     uint32Stat_t; /*< uint32 Stat type*/
692 BOOST_CLASS_VERSION(floatStat_t, 0)
693 BOOST_CLASS_VERSION(ucharStat_t, 0)
694 BOOST_CLASS_VERSION(uint16Stat_t, 0)
695 BOOST_CLASS_VERSION(uint32Stat_t, 0)



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
 * strictly prohibited
3  * and only allowed with the written consent of the author (
 * Jelle Spijker)
4  * This software is proprietary and confidential
5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6  */
7
8 #pragma once
9
10 #include "Stats.h"
11 #include <boost/serialization/base_object.hpp>
12
13 namespace SoilMath {
14 class PSD : public SoilMath::Stats<double, double, long
15     double> {
16     private:
17     uint32_t DetBin(float value) {
18         uint32_t i = noBins - 1;
19         while (i > 0) {
20             if (value > BinRanges[i]) {
21                 return i;
22             }
23             i--;

```

```

23     }
24     return 0;
25 }
26
27 void BasicCalculatePSD() {
28     float sum_dev = 0.0;
29     n = Rows * Cols;
30     for (uint32_t i = 0; i < n; i++) {
31         if (Data[i] > max) {
32             max = Data[i];
33         }
34         if (Data[i] < min) {
35             min = Data[i];
36         }
37         Sum += Data[i];
38     }
39     uint32_t index = 0;
40     Mean = Sum / (float)n;
41     Range = max - min;
42     for (uint32_t i = 0; i < n; i++) {
43         index = DetBin(Data[i]);
44         bins[index]++;
45         sum_dev += pow((Data[i] - Mean), 2);
46     }
47     Std = sqrt((float)(sum_dev / n));
48     getCFD();
49     Calculated = true;
50 }
51 friend class boost::serialization::access;
52
53 template <class Archive>
54 void serialize(Archive &ar, const unsigned int version) {
55     if (version == 0) {
56         ar &boost::serialization::base_object<
57             SoilMath::Stats<double, double, long double>(*
58             this);
59     }
60 }
61 public:
62 PSD() : SoilMath::Stats<double, double, long double>(){}
63
64 PSD(double *data, uint32_t nodata, double *binranges,
65      uint32_t nobins,
66      uint32_t endbin)
67      : SoilMath::Stats<double, double, long double>(nobins,
68          0, endbin) {
69     std::copy(binranges, binranges + nobins, BinRanges);
70     Data = data;
71     Rows = nodata;
72     Cols = 1;
73     BasicCalculatePSD();
74 }
75 }

```

```
76 BOOST_CLASS_VERSION(SoilMath::PSD, 0)
```

F.0.5 General project files

```

1  #-----
2  #
3  # Project created by QtCreator 2015-06-06T11:59:21
4  #
5  #-----
6
7  QT      += core gui concurrent
8  greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
9
10 TARGET = SoilMath
11 TEMPLATE = lib
12 VERSION = 0.9.8
13
14 DEFINES += SOILMATH_LIBRARY
15 QMAKE_CXXFLAGS += -std=c++11
16 unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../../../build/install/
17
18 @
19 CONFIG(release, debug|release):DEFINES += QT_NO_DEBUG_OUTPUT
20 @
21
22 SOURCES += \
23     NN.cpp \
24     GA.cpp \
25     FFT.cpp
26
27 HEADERS += \
28     Stats.h \
29     Sort.h \
30     SoilMathTypes.h \
31     SoilMath.h \
32     NN.h \
33     MathException.h \
34     GA.h \
35     FFT.h \
36     CommonOperations.h \
37     predict_t_archive.h \
38     Mat_archive.h \
39     psd.h
40
41 #opencv
42 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui
43 INCLUDEPATH += /usr/local/include/opencv
44 INCLUDEPATH += /usr/local/include
45
46 #boost
47 DEFINES += BOOST_ALL_DYN_LINK
48 INCLUDEPATH += /usr/include/boost
49 LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_serialization -lboost_iostreams
50
51 #Zlib
52 LIBS += -L/usr/local/lib -lz
53 INCLUDEPATH += /usr/local/include

```

```
54
55 unix {
56     target.path = $PWD/../../build/install
57     INSTALLS += target
58 }


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerspijker@gmail.com>, 2015
8 */
9
8 /*! \brief Collection of the public SoilMath headers
9  * Common practice is to include this header when you want to
10 * add Soilmath
11 */
12 #pragma once
13
14 #include "Stats.h"
15 #include "Sort.h"
16 #include "FFT.h"
17 #include "NN.h"
18 #include "GA.h"
19 #include "CommonOperations.h"
20 #include "SoilMathTypes.h"
21 #include "psd.h"
22 #include "Mat_archive.h"
23 #include "predict_t_archive.h"


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerspijker@gmail.com>, 2015
8 */
9
8 #pragma once
9 #define COMMONOPERATIONS_VERSION 1
10
11 #include <algorithm>
12 #include <stdint.h>
13 #include <math.h>
14 #include <vector>
15
16 namespace SoilMath {
17 inline uint16_t MinNotZero(uint16_t a, uint16_t b) {
18     if (a != 0 && b != 0) {
19         return (a < b) ? a : b;
20     } else {
21         return (a > b) ? a : b;
```

```

22     }
23 }
24
25 inline uint16_t Max(uint16_t a, uint16_t b) { return (a > b)
26     ? a : b; }
27
28 inline uint16_t Max(uint16_t a, uint16_t b, uint16_t c,
29     uint16_t d) {
30     return (Max(a, b) > Max(c, d)) ? Max(a, b) : Max(c, d);
31 }
32
33 inline uint16_t Min(uint16_t a, uint16_t b) { return (a < b)
34     ? a : b; }
35
36
37 static inline double quick_pow10(int n) {
38     static double pow10[19] = {1, 10, 100, 1000, 10000,
39     100000, 1000000, 10000000,
40     100000000, 1000000000, 10000000000,
41     100000000000, 1000000000000,
42     10000000000000, 100000000000000,
43     1000000000000000, 10000000000000000};
44 }
45
46
47 // Source:
48 // http://martin.ankerl.com/2012/01/25/optimized-
49 // approximative-pow-in-c-and-cpp/
50 static inline double fastPow(double a, double b) {
51     union {
52         double d;
53         int x[2];
54     } u = {a};
55     u.x[1] = (int)(b * (u.x[1] - 1072632447) + 1072632447);
56     u.x[0] = 0;
57     return u.d;
58 }
59
60 static inline double quick_pow2(int n) {
61     static double pow2[256] = {
62         0, 1, 4, 9, 16, 25, 36, 49,
63         64, 81,
64         100, 121, 144, 169, 196, 225, 256, 289,
65         324, 361,
66         400, 441, 484, 529, 576, 625, 676, 729,
67         784, 841,
68         900, 961, 1024, 1089, 1156, 1225, 1296, 1369,
69     }
70 }
```

```

1444, 1521,
65 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209,
2304, 2401,
66 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249,
3364, 3481,
67 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489,
4624, 4761,
68 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929,
6084, 6241,
69 6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569,
7744, 7921,
70 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409,
9604, 9801,
71 10000, 10201, 10404, 10609, 10816, 11025, 11236,
11449, 11664, 11881,
72 12100, 12321, 12544, 12769, 12996, 13225, 13456,
13689, 13924, 14161,
73 14400, 14641, 14884, 15129, 15376, 15625, 15876,
16129, 16384, 16641,
74 16900, 17161, 17424, 17689, 17956, 18225, 18496,
18769, 19044, 19321,
75 19600, 19881, 20164, 20449, 20736, 21025, 21316,
21609, 21904, 22201,
76 22500, 22801, 23104, 23409, 23716, 24025, 24336,
24649, 24964, 25281,
77 25600, 25921, 26244, 26569, 26896, 27225, 27556,
27889, 28224, 28561,
78 28900, 29241, 29584, 29929, 30276, 30625, 30976,
31329, 31684, 32041,
79 32400, 32761, 33124, 33489, 33856, 34225, 34596,
34969, 35344, 35721,
80 36100, 36481, 36864, 37249, 37636, 38025, 38416,
38809, 39204, 39601,
81 40000, 40401, 40804, 41209, 41616, 42025, 42436,
42849, 43264, 43681,
82 44100, 44521, 44944, 45369, 45796, 46225, 46656,
47089, 47524, 47961,
83 48400, 48841, 49284, 49729, 50176, 50625, 51076,
51529, 51984, 52441,
84 52900, 53361, 53824, 54289, 54756, 55225, 55696,
56169, 56644, 57121,
85 57600, 58081, 58564, 59049, 59536, 60025, 60516,
61009, 61504, 62001,
86 62500, 63001, 63504, 64009, 64516, 65025};
87  return pow2[(n >= 0) ? n : -n];
88 }
89
90 static inline long float2intRound(double d) {
91 d += 6755399441055744.0;
92 return reinterpret_cast<int &>(d);
93 }
94
95 /**
96 * \brief calcVolume according to ISO 9276-6
97 * \param A
98 * \return

```

```

99  /*
100 static inline float calcVolume(float A) {
101     return (pow(A, 1.5)) / 10.6347f;
102 }
103
104 static inline std::vector<float> makeOutput(uint8_t value,
105     uint32_t noNeurons) {
106     std::vector<float> retVal(noNeurons, -1);
107     retVal[value - 1] = 1;
108     return retVal;
109 }
110 */
111 * \brief calcDiameter according to ISO 9276-6
112 * \param A
113 * \return
114 */
115 static inline float calcDiameter(float A) {
116     //return sqrt((4 * A) / M_PI);
117     return 1.1283791670955 * sqrt(A);
118 }
119 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8  */
9 #pragma once
10
11 #define GENE_MAX 32 /*< maximum number of genes*/
12 #define CROSSOVER 16 /*< crossover location*/
13
14 #include <stdint.h>
15 #include <bitset>
16 #include <vector>
17 #include <complex>
18 #include <valarray>
19 #include <array>
20
21 typedef unsigned char uchar; /*< unsigned char*/
22 typedef unsigned short ushort; /*< unsigned short*/
23 typedef unsigned int uint32_t;
24
25 typedef std::complex<double> Complex_t; /*< complex
26  * vector of doubles*/
27 typedef std::vector<Complex_t> ComplexVect_t; /*< vector of
28  * Complex_t*/
29 typedef std::valarray<Complex_t> ComplexArray_t; /*<
30  * valarray of Complex_t*/
31 typedef std::vector<uint32_t> iContour_t; /*< vector
32  * of uint32_t*/
33 typedef std::bitset<GENE_MAX> Genome_t; /*< Bitset

```

```

    representing a genome*/
28 typedef std::pair<std::bitset<CROSSOVER>, std::bitset<
    GENE_MAX - CROSSOVER>>
29     SplitGenome_t; /*< a matted genome*/
30
31 typedef std::vector<float> Weight_t;      /*< a float vector
   */
32 typedef std::vector<Genome_t> GenVect_t; /*< a vector of
   genomes*/
33 typedef struct PopMemberStruct {
34     Weight_t weights;           /*< the weights the core of a
   population member*/
35     GenVect_t weightsGen;      /*< the weights as genomes*/
36     float Calculated = 0.0;    /*< the calculated value*/
37     float Fitness = 0.0;       /*< the fitness of the population
   member*/
38 } PopMember_t;                /*< a population member*/
39 typedef std::vector<PopMember_t> Population_t; /*< Vector
   with PopMember_t*/
40 typedef std::pair<float, float>
41     MinMaxWeight_t; /*< floating pair weight range*/
42
43 typedef struct Predict_struct {
44     uint8_t Category = 1; /*< the category number */
45     float RealValue = 1.; /*< category number as float in
   order to estimate how
   precise to outcome is*/
46     float Accuracy = 1.; /*< the accuracy of the category*/
47     std::vector<float> OutputNeurons; /*< the output Neurons
   */
48     bool ManualSet = true;
49 } Predict_t;                  /*< The prediction
   results*/
50
51 typedef Predict_t (*NNfunctionType)(
52     ComplexVect_t, Weight_t, Weight_t, uint32_t, uint32_t,
53     uint32_t); /*< The prediction function from the Neural
   Net*/
54
55 typedef std::vector<ComplexVect_t>
56     InputLearnVector_t; /*< Vector of a vector with complex
   values*/
57 typedef std::vector<Predict_t> OutputLearnVector_t; /*<
   vector with results*/

```

```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
   strictly prohibited
3  * and only allowed with the written consent of the author (
   Jelle Spijker)
4  * This software is proprietary and confidential
5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6  */
7
8 // Source:
9 // http://stackoverflow.com/questions/16125574/how-to-
   serialize-opencv-mat-with-boost-xml-archive

```

```

10 #pragma once
11
12 #include <boost/archive/binary_iarchive.hpp>
13 #include <boost/archive/binary_oarchive.hpp>
14 #include <boost/serialization/access.hpp>
15 #include <opencv/cv.h>
16 #include <opencv2/core.hpp>
17
18 namespace boost {
19 namespace serialization {
20 /*!
21 * \brief serialize Serialize the openCV mat to disk
22 */
23 template <class Archive>
24 inline void serialize(Archive &ar, cv::Mat &m, const
25           unsigned int version __attribute__((unused))) {
26     int cols = m.cols;
27     int rows = m.rows;
28     int elemSize = m.elemSize();
29     int elemType = m.type();
30
31     ar &cols;
32     ar &rows;
33     ar &elemSize;
34     ar &elemType; // element type.
35
36     if (m.type() != elemType || m.rows != rows || m.cols !=
37         cols) {
38         m = cv::Mat(rows, cols, elemType, cv::Scalar(0));
39     }
40
41     size_t dataSize = cols * rows * elemSize;
42
43     for (size_t dc = 0; dc < dataSize; dc++) {
44         ar &m.data[dc];
45     }
46 }

```

```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9 // Source:
10 // http://stackoverflow.com/questions/16125574/how-to-
11 // serialize-opencv-mat-with-boost-xml-archive
12 #pragma once
13 #include <boost/archive/binary_iarchive.hpp>
14 #include <boost/archive/binary_oarchive.hpp>

```

```

14 #include <boost/serialization/access.hpp>
15 #include <boost/serialization/vector.hpp>
16 #include <boost/serialization/complex.hpp>
17 #include "SoilMathTypes.h"
18
19 namespace boost {
20 namespace serialization {
21 /*!
22 * \brief serialize Serialize the openCV mat to disk
23 */
24 template <class Archive>
25 inline void serialize(Archive &ar, Predict_t &P, const
26     unsigned int version __attribute__((unused))) {
27     ar &P.Accuracy;
28     ar &P.Category;
29     ar &P.OutputNeurons;
30     ar &P.RealValue;
31 }
32 }

/*
Copyright (C) Jelle Spijker - All Rights Reserved
* Unauthorized copying of this file, via any medium is
strictly prohibited
* and only allowed with the written consent of the author (
Jelle Spijker)
* This software is proprietary and confidential
* Written by Jelle Spijker <spijkerspijker@gmail.com>, 2015
*/
7
8 #define EXCEPTION_MATH "Math Exception!"
9 #define EXCEPTION_MATH_NR 0
10 #define EXCEPTION_NO_CONTOUR_FOUND
11     "No continuous contour found, or less then 8 pixels long!"
12 #define EXCEPTION_NO_CONTOUR_FOUND_NR 1
13 #define EXCEPTION_SIZE_OF_INPUT_NEURONS
14     "Size of input unequal to input neurons exception!"
15 #define EXCEPTION_SIZE_OF_INPUT_NEURONS_NR 2
16 #define EXCEPTION_NEURAL_NET_NOT_STUDIED "Neural net didn't
study exception!"
17 #define EXCEPTION_NEURAL_NET_NOT_STUDIED_NR 3
18 #define EXCEPTION_TYPE_NOT_SUPPORTED
19     "Type not supported for operation exception!"
20 #define EXCEPTION_TYPE_NOT_SUPPORTED_NR 4
21
22 #pragma once
23 #include <exception>
24 #include <string>
25
26 namespace SoilMath {
27 namespace Exception {
28 class MathException : public std::exception {
29 public:

```



```
37         r--;
38     } else {
39         T t = *l;
40         *l = *r;
41         *r = t;
42         l++;
43         r--;
44     }
45 }
46 Sort::QuickSort<T>(arr, r - arr + 1);
47 Sort::QuickSort<T>(l, arr + i - 1);
48 }
49
50 /**
51 * \brief QuickSort a static sort a Type T array with i
52 *        values where the key
53 *        are also changed accordingly
54 * \details Usage: QuickSort<type>(*type *type , i)
55 * \param arr an array of Type T
56 * \param key an array of 0..i-1 representing the index
57 * \param i the number of elements
58 */
59 template <typename T> static void QuickSort(T *arr, T *key
60     , int i) {
61     if (i < 2)
62         return;
63
64     T p = arr[i / 2];
65
66     T *l = arr;
67     T *r = arr + i - 1;
68
69     T *lkey = key;
70     T *rkey = key + i - 1;
71
72     while (l <= r) {
73         if (*l < p) {
74             l++;
75             lkey++;
76         } else if (*r > p) {
77             r--;
78             rkey--;
79         } else {
80             if (*l != *r) {
81                 T t = *l;
82                 *l = *r;
83                 *r = t;
84
85                 T tkey = *lkey;
86                 *lkey = *rkey;
87                 *rkey = tkey;
88             }
89
90             l++;
91             r--;
92         }
93     }
94 }
```

```
91         lkey++;
92         rkey--;
93     }
94 }
95 Sort::QuickSort<T>(arr, key, r - arr + 1);
96 Sort::QuickSort<T>(l, lkey, arr + i - 1);
97 }
98 }
99 }
```



G. Hardware Library

G.0.6 Microscope Class

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 /*! \class Microscope
11 Interaction with the USB 5 MP microscope
12 */
13
14 #pragma once
15
16 #include <stdint.h>
17 #include <vector>
18 #include <string>
19 #include <utility>
20 #include <algorithm>
21
22 #include <sys/stat.h>
23 #include <sys/utsname.h>
24 #include <sys/ioctl.h>
25 #include <fstream>
26 #include <fcntl.h>
27
28 #include <linux/videodev2.h>
29 #include <linux/v4l2-controls.h>
30 #include <linux/v4l2-common.h>
```

```
30 #include <boost/filesystem.hpp>
31 #include <boost/regex.hpp>
32
33 #include <opencv2/photo.hpp>
34 #include <opencv2/imgcodecs.hpp>
35 #include <opencv2/opencv.hpp>
36 #include <opencv2/core.hpp>
37
38 #include "MicroscopeNotFoundException.h"
39 #include "CouldNotGrabImageException.h"
40
41 namespace Hardware {
42 class Microscope {
43 public:
44     enum Arch { ARM, X64 };
45
46     enum PixelFormat { YUYV, MJPG };
47
48     struct Resolution_t {
49         uint16_t Width = 2048;
50         uint16_t Height = 1536;
51         PixelFormat format = PixelFormat::MJPEG;
52         std::string to_string() {
53             std::string retVal = std::to_string(Width);
54             retVal.append(" x ");
55             retVal.append(std::to_string(Height));
56             if (format == PixelFormat::MJPEG) {
57                 retVal.append(" - MJPG");
58             }
59             else {
60                 retVal.append(" - YUYV");
61             }
62             return retVal;
63         }
64         uint32_t ID;
65     };
66
67     struct Control_t {
68         std::string name;
69         int minimum;
70         int maximum;
71         int step;
72         int default_value;
73         int current_value;
74         uint32_t ID = V4L2_CID_BASE;
75         bool operator==(Control_t &rhs) {
76             if (this->name.compare(rhs.name) == 0) {
77                 return true;
78             } else {
79                 return false;
80             }
81         }
82         bool operator!=(Control_t &rhs) {
83             if (this->name.compare(rhs.name) != 0) {
84                 return true;
85             } else {
```

```
86         return false;
87     }
88 }
89 };
90
91 typedef std::vector<Control_t> Controls_t;
92
93 struct Cam_t {
94     std::string Name;
95     std::string devString;
96     uint32_t ID;
97     std::vector<Resolution_t> Resolutions;
98     uint32_t delaytrigger = 1;
99     Resolution_t *SelectedResolution = nullptr;
100    Controls_t Controls;
101    int fd;
102    bool operator==(Cam_t const &rhs) {
103        if (this->ID == rhs.ID || this->Name == rhs.Name) {
104            return true;
105        } else {
106            return false;
107        }
108    }
109    bool operator!=(Cam_t const &rhs) {
110        if (this->ID != rhs.ID && this->Name != rhs.Name) {
111            return true;
112        } else {
113            return false;
114        }
115    }
116 };
117
118 std::vector<Cam_t> AvailableCams;
119 Cam_t *SelectedCam = nullptr;
120 Arch RunEnv;
121
122 Microscope();
123 Microscope(const Microscope &rhs);
124
125 ~Microscope();
126
127 Microscope operator=(Microscope const &rhs);
128
129 bool IsOpened();
130 bool openCam(Cam_t *cam);
131 bool openCam(int &cam);
132 bool openCam(std::string &cam);
133
134 bool closeCam(Cam_t *cam);
135
136 void GetFrame(cv::Mat &dst);
137 void GetHDRFrame(cv::Mat &dst, uint32_t noframes = 3);
138
139 Control_t *GetControl(const std::string name);
140 void SetControl(Control_t *control);
141
```

```

142     Cam_t *FindCam(std::string cam);
143     Cam_t *FindCam(int cam);
144
145 private:
146     cv::VideoCapture *cap = nullptr;
147
148     std::vector<cv::Mat> HDRframes;
149
150     std::vector<Cam_t> GetAvailableCams();
151     Arch GetCurrentArchitecture();
152     int fd;
153 };
154 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (Jelle Spijker)
5  * This software is proprietary and confidential
6  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
7  */
8
9 #include "Microscope.h"
10
11 namespace Hardware {
12
13     Microscope::Microscope() {
14         RunEnv = GetCurrentArchitecture();
15         AvailableCams = GetAvailableCams();
16         for_each(AvailableCams.begin(), AvailableCams.end(), [](
17             Cam_t &C) {
18             C.SelectedResolution = &C.Resolutions[C.Resolutions.size
19             () - 1];
20         });
21     }
22
23     Microscope::Microscope(const Microscope &rhs) {
24         std::copy(rhs.AvailableCams.begin(), rhs.AvailableCams.end
25         (),
26             this->AvailableCams.begin());
27         this->RunEnv = rhs.RunEnv;
28         this->SelectedCam = rhs.SelectedCam;
29         this->cap = rhs.cap;
30         this->fd = rhs.fd;
31         this->HDRframes = rhs.HDRframes;
32     }
33
34     Microscope::~Microscope() { delete cap; }
35
36     Microscope::Arch Microscope::GetCurrentArchitecture() {
37         struct utsname unameData;
38         Arch retVal;
39         uname(&unameData);
40         std::string archString = static_cast<std::string>(
41             unameData.machine);

```

```

37     if (archString.find("armv7l") != string::npos) {
38         retVal = Arch::ARM;
39     } else {
40         retVal = Arch::X64;
41     }
42     return retVal;
43 }
44
45 std::vector<Microscope::Cam_t> Microscope::GetAvailableCams
46 () {
47     const string path_ss = "/sys/class/video4linux";
48     const string path_ss_dev = "/dev/video";
49     std::vector<Cam_t> retVal;
50     struct v4l2_queryctrl queryctrl;
51     struct v4l2_control controlctrl;
52
53     // Check if there're videodevices installed
54     // Iterate through the cams
55     for (boost::filesystem::directory_iterator itr(path_ss);
56          itr != boost::filesystem::directory_iterator(); ++itr
57          ) {
58         string videoLn = itr->path().string();
59         videoLn.append("/name");
60         if (boost::filesystem::exists(videoLn)) {
61             Cam_t currentCam;
62             std::ifstream camName;
63             camName.open(videoLn);
64             std::getline(camName, currentCam.Name);
65             camName.close();
66             currentCam.ID =
67                 std::atoi(itr->path().string().substr(28, std::
68                           string::npos).c_str());
69
70             // Open Cam
71             currentCam.devString = path_ss_dev + std::to_string(
72                 currentCam.ID);
73             if ((currentCam.fd = open(currentCam.devString.c_str(),
74                 O_RDWR)) == -1) {
75                 throw Exception::MicroscopeException(
76                     EXCEPTION_NOCAMS,
77                     EXCEPTION_NOCAMS_NR
78                 );
79
80             // Get controls
81             memset(&queryctrl, 0, sizeof(queryctrl));
82             memset(&controlctrl, 0, sizeof(controlctrl));
83             for (queryctrl.id = V4L2_CID_BASE; queryctrl.id <
84                 V4L2_CID_LASTP1;
85                 queryctrl.id++) {
86
87                 if (ioctl(currentCam.fd, VIDIOC_QUERYCTRL, &
88                     queryctrl) == 0) {
89                     if (!(queryctrl.flags & V4L2_CTRL_FLAG_DISABLED))
90                     {
91                         Control_t currentControl;

```

```

83         currentControl.ID = queryctrl.id;
84         currentControl.name = (char *)queryctrl.name;
85         currentControl.minimum = queryctrl.minimum;
86         currentControl.maximum = queryctrl.maximum;
87         currentControl.default_value = queryctrl.
88             default_value;
89         currentControl.step = queryctrl.step;
90         controlctrl.id = queryctrl.id;
91         if (ioctl(currentCam.fd, VIDIOC_G_CTRL, &
92             controlctrl) == 0) {
93             currentControl.current_value = controlctrl.
94                 value;
95         }
96     } else {
97         if (errno == EINVAL)
98             continue;
99         throw Exception::MicroscopeException(
100             EXCEPTION_QUERY,
101             EXCEPTION_QUERY_NR
102             );
103     }
104 }
105
106 // Get image formats
107 struct v4l2_format format;
108 memset(&format, 0, sizeof(format));
109
110 uint32_t width[5] = {640, 800, 1280, 1600, 2048};
111 uint32_t height[6] = {480, 600, 960, 1200, 1536, 1920};
112
113 uint32_t ResolutionID = 0;
114
115 // YUYV
116 for (uint32_t i = 0; i < 5; i++) {
117     format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
118     format.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;
119     format.fmt.pix.width = width[i];
120     format.fmt.pix.height = height[i];
121     int ret = ioctl(currentCam.fd, VIDIOC_S_FMT, &format
122     );
123     if (ret != -1 && format.fmt.pix.height == height[i]
124         &&
125             format.fmt.pix.width == width[i]) {
126         Resolution_t res;
127         res.Width = format.fmt.pix.width;
128         res.Height = height[i];
129         res.ID = ResolutionID++;
130         res.format = PixelFormat::YUYV;
131         currentCam.Resolutions.push_back(res);
132     }
133 }
134
135 // MJPEG
136 for (uint32_t i = 0; i < 5; i++) {

```

```

132     format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
133     format.fmt.pix.pixelformat = V4L2_PIX_FMT_MJPEG;
134     format.fmt.pix.width = width[i];
135     format.fmt.pix.height = height[i];
136     int ret = ioctl(currentCam.fd, VIDIOC_S_FMT, &format
137     );
138     if (ret != -1 && format.fmt.pix.height == height[i]
139     &&
140         format.fmt.pix.width == width[i]) {
141         Resolution_t res;
142         res.Width = format.fmt.pix.width;
143         res.Height = format.fmt.pix.height;
144         res.ID = ResolutionID++;
145         res.format = PixelFormat::MJPEG;
146         currentCam.Resolutions.push_back(res);
147     }
148     close(currentCam.fd);
149     retVal.push_back(currentCam);
150 }
151 }
152
153 for (uint32_t i = 0; i < retVal.size(); i++) {
154     if (retVal[i].Resolutions.size() == 0) {
155         retVal.erase(retVal.begin() + i);
156         i--;
157     }
158 }
159
160     return retVal;
161 }
162
163 bool Microscope::IsOpened() {
164     if (cap == nullptr) {
165         return false;
166     } else {
167         return cap->isOpened();
168     }
169 }
170
171 bool Microscope::openCam(Cam_t *cam) {
172     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
173         if (AvailableCams[i] == *cam) {
174             closeCam(SelectedCam);
175             SelectedCam = cam;
176             cap = new cv::VideoCapture(SelectedCam->ID);
177             if (!cap->isOpened()) {
178                 throw Exception::MicroscopeException(
179                     EXCEPTION_NOCAMS,
180                     EXCEPTION_NOCAMS_NR
181                     );
182             }
183             cap->set(CV_CAP_PROP_FRAME_WIDTH, SelectedCam->
184                 SelectedResolution->Width);
185             cap->set(CV_CAP_PROP_FRAME_HEIGHT,

```

```
183         SelectedCam->SelectedResolution->Height);
184     for (Controls_t::iterator it = SelectedCam->Controls.
185         begin();
186         it != SelectedCam->Controls.end(); ++it) {
187         SetControl(&*it);
188     }
189 }
190     return true;
191 }
192 }
193
194 bool Microscope::openCam(std::string &cam) { return openCam(
195     FindCam(cam)); }
196
197 bool Microscope::openCam(int &cam) { return openCam(FindCam(
198     cam)); }
199
200 Microscope::Cam_t *Microscope::FindCam(int cam) {
201     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
202         if (cam == AvailableCams[i].ID) {
203             return &AvailableCams[i];
204         }
205     }
206     return nullptr;
207 }
208
209 Microscope::Cam_t *Microscope::FindCam(string cam) {
210     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
211         if (cam.compare(AvailableCams[i].Name) == 0) {
212             return &AvailableCams[i];
213         }
214     }
215     return nullptr;
216 }
217
218 bool Microscope::closeCam(Cam_t *cam) {
219     if (cap != nullptr) {
220         if (cap->isOpened()) {
221             cap->release();
222         }
223         delete cap;
224         cap = nullptr;
225     }
226 }
227
228 void Microscope::GetFrame(cv::Mat &dst) {
229     openCam(SelectedCam);
230     sleep(SelectedCam->delaytrigger);
231     if (RunEnv == Arch::ARM) {
232         for (uint32_t i = 0; i < 2; i++) {
233             if (!cap->grab()) {
234                 throw Exception::CouldNotGrabImageException();
235             }
236             sleep(SelectedCam->delaytrigger);
237         }
238     }
239 }
```

```

236     cap->retrieve(dst);
237 } else {
238     for (uint32_t i = 0; i < 2; i++) {
239         if (!cap->read(dst)) {
240             throw Exception::CouldNotGrabImageException();
241         }
242     }
243 }
244 }
245
246 void Microscope::GetHDRFrame(cv::Mat &dst, uint32_t noframes
247 ) {
248 // create the brightness steps
249 Control_t *brightness = GetControl("Brightness");
250 Control_t *contrast = GetControl("Contrast");
251
252     uint32_t brightnessStep =
253         (brightness->maximum - brightness->minimum) / noframes
254         ;
255     int8_t currentBrightness = brightness->current_value;
256     int8_t currentContrast = contrast->current_value;
257     contrast->current_value = contrast->maximum;
258
259     cv::Mat currentImg;
260     // take the shots at different brightness levels
261     for (uint32_t i = 1; i <= noframes; i++) {
262         brightness->current_value = brightness->minimum + (i *
263             brightnessStep);
264         GetFrame(currentImg);
265         HDRframes.push_back(currentImg);
266     }
267
268     // Set the brightness and back to the previous used level
269     brightness->current_value = currentBrightness;
270     contrast->current_value = currentContrast;
271
272     // Perform the exposure fusion
273     cv::Mat fusion;
274     cv::Ptr<cv::MergeMertens> merge_mertens = cv::
275         createMergeMertens();
276     merge_mertens->process(HDRframes, fusion);
277     fusion *= 255;
278     fusion.convertTo(dst, CV_8UC1);
279 }
280
281 Microscope::Control_t *Microscope::GetControl(const string
282 name) {
283     for (Controls_t::iterator it = SelectedCam->Controls.begin
284         ());
285         it != SelectedCam->Controls.end(); ++it) {
286         if (name.compare(it->name) == 0) {
287             return &*it;
288         }
289     }
290 }
291
292 return nullptr;
293 }
```

```

286
287 void Microscope::SetControl(Control_t *control) {
288     if ((SelectedCam->fd = open(SelectedCam->devString.c_str()
289         , O_RDWR)) == -1) {
290         throw Exception::MicroscopeException(EXCEPTION_NOCAMS,
291             EXCEPTION_NOCAMS_NR);
292     }
293
294     struct v4l2_queryctrl queryctrl;
295     struct v4l2_control controlctrl;
296
297     memset(&queryctrl, 0, sizeof(queryctrl));
298     queryctrl.id = control->ID;
299     if (ioctl(SelectedCam->fd, VIDIOC_QUERYCTRL, &queryctrl)
300         == -1) {
301         if (errno != EINVAL) {
302             close(SelectedCam->fd);
303             throw Exception::MicroscopeException(EXCEPTION_QUERY,
304                 EXCEPTION_QUERY_NR);
305         } else {
306             close(SelectedCam->fd);
307             throw Exception::MicroscopeException(
308                 EXCEPTION_CTRL_NOT_FOUND,
309                 EXCEPTION_CTRL_NOT_FOUND_NR
310             );
311         }
312     } else if (queryctrl.flags & V4L2_CTRL_FLAG_DISABLED) {
313         close(SelectedCam->fd);
314         throw Exception::MicroscopeException(
315             EXCEPTION_CTRL_NOT_FOUND,
316             EXCEPTION_CTRL_NOT_FOUND_NR
317         );
318     }
319     }
320 }
321     close(SelectedCam->fd);
322 }
323 }
```

G.0.7 Beaglebone Black Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10  /*! \class BBB
11  The core BeagleBone Black class used for all hardware
12  related classes.
13  Consisting of universal used method, functions and variables
14  . File operations,
15  polling and threading
16  */
17
18
19  #pragma once
20
21  #define SLOTS
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```

```

46
47 protected:
48     bool threadRunning;           /*!< used to stop the
49     thread*/
50     pthread_t thread;           /*!< The thread*/
51     CallbackType callbackFunction; /*!< the callbackfunction*/
52
53     bool DirectoryExist(const string &path);
54     bool CapeLoaded(const string &shield);
55
56     string Read(const string &path);
57     void Write(const string &path, const string &value);
58
59     /*! Converts a number to a string
60     \param Number as typename
61     \returns the number as a string
62     */
63     template <typename T> string NumberToString(T Number) {
64         ostringstream ss;
65         ss << Number;
66         return ss.str();
67     }
68
69     /*! Converts a string to a number
70     \param Text the string that needs to be converted
71     \return the number as typename
72     */
73     template <typename T> T StringToNumber(string Text) {
74         stringstream ss(Text);
75         T result;
76         return ss >> result ? result : 0;
77     };
78 }



---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (Jelle Spijker)
5  * This software is proprietary and confidential
6  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
7  */
8
9
10 namespace Hardware {
11 /*! Constructor*/
12 BBB::BBB() {
13     threadRunning = false;
14     callbackFunction = NULL;
15     debounceTime = 0;
16     thread = (pthread_t)NULL;
17 }
18
19 /*! De-constructor*/

```

```

20 BBB::~BBB() {}
21
22 /*! Reads the first line from a file
23 \param path constant string pointing towards the file
24 \returns this first line
25 */
26 string BBB::Read(const string &path) {
27     ifstream fs;
28     fs.open(path.c_str());
29     if (!fs.is_open()) {
30         throw Exception::GPIOReadException("Can't open: " +
31             path).c_str());
32     }
33     string input;
34     getline(fs, input);
35     fs.close();
36     return input;
37 }
38 /*! Writes a value to a file
39 \param path a constant string pointing towards the file
40 \param value a constant string which should be written in
41     the file
42 */
43 void BBB::Write(const string &path, const string &value) {
44     ofstream fs;
45     fs.open(path.c_str());
46     if (!fs.is_open()) {
47         throw Exception::GPIOReadException("Can't open: " +
48             path).c_str());
49     }
50     fs << value;
51     fs.close();
52 }
53 /*! Checks if a directory exist
54 \returns true if the directory exists and false if not
55 */
56 bool BBB::DirectoryExist(const string &path) {
57     struct stat st;
58     if (stat((char *)path.c_str(), &st) != 0) {
59         return false;
60     }
61     return true;
62 }
63 /*! Checks if a cape is loaded in the file /sys/devices/
64     bone_capemgr.9/slots
65 \param shield a const search string which is a (part) of the
66     shield name
67 \return true if the search string is found otherwise false
68 */
69 bool BBB::CapeLoaded(const string &shield) {
70     bool shieldFound = false;
71
72     ifstream fs;
73     string line;
74
75     fs.open(path.c_str());
76     if (!fs.is_open()) {
77         throw Exception::GPIOReadException("Can't open: " +
78             path).c_str());
79     }
80
81     while (getline(fs, line)) {
82         if (line.find(shield) != string::npos) {
83             shieldFound = true;
84             break;
85         }
86     }
87
88     fs.close();
89
90     return shieldFound;
91 }
```

```
71     fs.open(SLOTS);
72     if (!fs.is_open()) {
73         throw Exception::GPIOReadException("Can't open SLOTS");
74     }
75
76     string line;
77     while (getline(fs, line)) {
78         if (line.find(shield) != string::npos) {
79             shieldFound = true;
80             break;
81         }
82     }
83     fs.close();
84     return shieldFound;
85 }
86 }
```

G.0.8 GPIO Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   * This code is based upon:
9   * Derek Molloy, "Exploring BeagleBone: Tools and Techniques
10  * for Building
11  * with Embedded Linux", Wiley, 2014, ISBN:9781118935125.
12  * See: www.exploringbeaglebone.com
13  */
14
15 #pragma once
16 #include "BBB.h"
17
18 #define EXPORT_PIN "/sys/class/gpio/export"
19 #define UNEXPORT_PIN "/sys/class/gpio/unexport"
20 #define GPIOS "/sys/class/gpio/gpio"
21 #define DIRECTION "/direction"
22 #define VALUE "/value"
23 #define EDGE "/edge"
24
25 using namespace std;
26
27 namespace Hardware {
28 class GPIO : public BBB {
29 public:
30     enum Direction { Input, Output };
31     enum Value { Low = 0, High = 1 };
32     enum Edge { None, Rising, Falling, Both };
33     int number; // Number of the pin
34     int WaitForEdge();
35     int WaitForEdge(CallbackType callback);
36     void WaitForEdgeCancel() { this->threadRunning = false; }
37     Value GetValue();
38     void SetValue(Value value);
39     Direction GetDirection();
40     void SetDirection(Direction direction);
41     Edge GetEdge();
42     void SetEdge(Edge edge);
43     GPIO(int number);
44     ~GPIO();
45
46 private:
47     string gpiopath;
48     Direction direction;

```

```

52     Edge edge;
53     friend void *threadedPollGPIO(void *value);
54
55     bool isExported(int number, Direction &dir, Edge &edge);
56     bool ExportPin(int number);
57     bool UnexportPin(int number);
58
59     Direction ReadsDirection(const string &gpiopath);
60     void WritesDirection(const string &gpiopath, Direction
61                           direction);
62
63     Edge ReadsEdge(const string &gpiopath);
64     void WritesEdge(const string &gpiopath, Edge edge);
65
66     Value ReadsValue(const string &gpiopath);
67     void WritesValue(const string &gpiopath, Value value);
68 };
69 void *threadedPollGPIO(void *value);
70 }
```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #include "GPIO.h"
11
12 namespace Hardware {
13 GPIO::GPIO(int number) {
14
15     this->number = number;
16     gpiopath = GPIOs + NumberToString<int>(number);
17
18     if (!isExported(number, direction, edge)) {
19         ExportPin(number);
20         direction = ReadsDirection(gpiopath);
21         edge = ReadsEdge(gpiopath);
22     }
23     usleep(250000);
24 }
25
26 GPIO::~GPIO() { UnexportPin(number); }
27
28 int GPIO::WaitForEdge(CallbackType callback) {
29     threadRunning = true;
30     callbackFunction = callback;
31     if (pthread_create(&this->thread, NULL, &threadedPollGPIO,
32                       static_cast<void *>(this))) {
33         threadRunning = false;
34         throw Exception::
35             FailedToCreateGPIOPollingThreadException();
36     }
37 }
```

```

33     }
34     return 0;
35 }
36
37 int GPIO::WaitForEdge() {
38     if (direction == Output) {
39         SetDirection(Input);
40     }
41     int fd, i, epollfd, count = 0;
42     struct epoll_event ev;
43     epollfd = epoll_create(1);
44     if (epollfd == -1) {
45         throw Exception::
46             FailedToCreateGPIOPollingThreadException(
47                 "GPIO: Failed to create epollfd!");
48     }
49     if ((fd = open((gpiopath + VALUE).c_str(), O_RDONLY |
50                     O_NONBLOCK)) == -1) {
51         throw Exception::GPIOReadException();
52     }
53
54     // read operation | edge triggered | urgent data
55     ev.events = EPOLLIN | EPOLLET | EPOLLPRI;
56     ev.data.fd = fd;
57
58     if (epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev) == -1) {
59         throw Exception::
60             FailedToCreateGPIOPollingThreadException(
61                 "GPIO: Failed to add control interface!");
62     }
63
64     while (count <= 1) {
65         i = epoll_wait(epollfd, &ev, 1, -1);
66         if (i == -1) {
67             close(fd);
68             return -1;
69         } else {
70             count++;
71         }
72     }
73     close(fd);
74     return 0;
75 }
76
77 GPIO::Value GPIO::GetValue() { return ReadsValue(gpiopath);
78 }
79
80 void GPIO::SetValue(GPIO::Value value) { WritesValue(
81     gpiopath, value); }
82
83 GPIO::Direction GPIO::GetDirection() { return direction; }
84 void GPIO::SetDirection(Direction direction) {
85     this->direction = direction;
86     WritesDirection(gpiopath, direction);
87 }
88
89 GPIO::Edge GPIO::GetEdge() { return edge; }

```

```
84 void GPIO::SetEdge(Edge edge) {
85     this->edge = edge;
86     WritesEdge(gpiopath, edge);
87 }
88
89 bool GPIO::isExported(int number __attribute__((unused)),
90     Direction &dir,
91     Edge &edge) {
92     // Checks if directory exist and therefore is exported
93     if (!DirectoryExist(gpiopath)) {
94         return false;
95     }
96     // Reads the data associated with the pin
97     dir = ReadsDirection(gpiopath);
98     edge = ReadsEdge(gpiopath);
99     return true;
100 }
101
102 bool GPIO::ExportPin(int number) {
103     switch (number) {
104     case 7:
105         system("config-pin P9.42 gpio");
106         break;
107     case 116:
108         system("config-pin P9.91 gpio");
109         break;
110     case 112:
111         system("config-pin P9.30 gpio");
112         break;
113     case 115:
114         system("config-pin P9.27 gpio");
115         break;
116     case 14:
117         system("config-pin P9.26 gpio");
118         break;
119     case 15:
120         system("config-pin P9.24 gpio");
121         break;
122     case 49:
123         system("config-pin P9.23 gpio");
124         break;
125     case 2:
126         system("config-pin P9.22 gpio");
127         break;
128     case 3:
129         system("config-pin P9.21 gpio");
130         break;
131     case 4:
132         system("config-pin P9.18 gpio");
133         break;
134     case 5:
135         system("config-pin P9.17 gpio");
136         break;
137     case 51:
138         system("config-pin P9.16 gpio");
```

```
139     break;
140 case 48:
141     system("config-pin P9.15 gpio");
142     break;
143 case 50:
144     system("config-pin P9.14 gpio");
145     break;
146 case 31:
147     system("config-pin P9.13 gpio");
148     break;
149 case 60:
150     system("config-pin P9.12 gpio");
151     break;
152 case 30:
153     system("config-pin P9.11 gpio");
154     break;
155 case 61:
156     system("config-pin P8.26 gpio");
157     break;
158 case 22:
159     system("config-pin P8.19 gpio");
160     break;
161 case 65:
162     system("config-pin P8.18 gpio");
163     break;
164 case 27:
165     system("config-pin P8.17 gpio");
166     break;
167 case 46:
168     system("config-pin P8.16 gpio");
169     break;
170 case 47:
171     system("config-pin P8.15 gpio");
172     break;
173 case 26:
174     system("config-pin P8.14 gpio");
175     break;
176 case 23:
177     system("config-pin P8.13 gpio");
178     break;
179 case 44:
180     system("config-pin P8.12 gpio");
181     break;
182 case 45:
183     system("config-pin P8.11 gpio");
184     break;
185 case 68:
186     system("config-pin P8.10 gpio");
187     break;
188 case 69:
189     system("config-pin P8.09 gpio");
190     break;
191 case 67:
192     system("config-pin P8.08 gpio");
193     break;
194 case 66:
```

```
195     system("config-pin P8.07 gpio");
196     break;
197 }
198 usleep(250000);
199 }
200
201 bool GPIO::UnexportPin(int number) {
202     //Write(UNEXPORT_PIN, NumberToString<int>(number));
203 }
204
205 GPIO::Direction GPIO::ReadsDirection(const string &gpiopath)
206 {
207     if (Read(gpiopath + DIRECTION) == "in") {
208         return Input;
209     } else {
210         return Output;
211     }
212 }
213
214 void GPIO::WritesDirection(const string &gpiopath, Direction
215     direction) {
216     switch (direction) {
217     case Hardware::GPIO::Input:
218         Write((gpiopath + DIRECTION), "in");
219         break;
220     case Hardware::GPIO::Output:
221         Write((gpiopath + DIRECTION), "out");
222         break;
223     }
224 }
225
226 GPIO::Edge GPIO::ReadsEdge(const string &gpiopath) {
227     string reader = Read(gpiopath + EDGE);
228     if (reader == "none") {
229         return None;
230     } else if (reader == "rising") {
231         return Rising;
232     } else if (reader == "falling") {
233         return Falling;
234     } else {
235         return Both;
236     }
237 }
238
239 void GPIO::WritesEdge(const string &gpiopath, Edge edge) {
240     switch (edge) {
241     case Hardware::GPIO::None:
242         Write((gpiopath + EDGE), "none");
243         break;
244     case Hardware::GPIO::Rising:
245         Write((gpiopath + EDGE), "rising");
246         break;
247     case Hardware::GPIO::Falling:
248         Write((gpiopath + EDGE), "falling");
249         break;
250     case Hardware::GPIO::Both:
```

```
249     Write((gpiopath + EDGE), "both");
250     break;
251     default:
252     break;
253 }
254 }
255
256 GPIO::Value GPIO::ReadsValue(const string &gpiopath) {
257     string path(gpiopath + VALUE);
258     int res = StringToNumber<int>(Read(path));
259     return (Value)res;
260 }
261
262 void GPIO::WritesValue(const string &gpiopath, Value value)
263 {
264     Write(gpiopath + VALUE, NumberToString<int>(value));
265 }
266
267 void *threadedPollGPIO(void *value) {
268     GPIO *gpio = static_cast<GPIO *>(value);
269     while (gpio->threadRunning) {
270         gpio->callbackFunction(gpio->WaitForEdge());
271         usleep(gpio->debounceTime * 1000);
272     }
273 }
274 }
```

G.0.9 PWM Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11 #include "BBB.h"
12 #include <dirent.h>
13
14 #define OCP_PATH "/sys/class/pwm/"
15 #define PWM_CAPE "Override Board Name,00A0,Override Manuf,
16   cape-universal"
17
18 namespace Hardware {
19 class PWM : public BBB {
20 public:
21   enum Pin // Four possible PWM pins
22   {
23     P8_13,
24     P8_19,
25     P9_14,
26     P9_16
27   };
28   enum Run // Signal generating
29   {
30     On = 1,
31     Off = 0
32   };
33   enum Polarity // Inverse duty polarity
34   {
35     Normal = 1,
36     Inverted = 0
37   };
38   Pin pin; // Current pin
39
40   uint8_t GetPixelValue() { return pixelvalue; }
41   void SetPixelValue(uint8_t value);
42
43   float GetIntensity() { return intensity; }
44   void SetIntensity(float value);
45
46   int GetPeriod() { return period; }
47   void SetPeriod(int value);
48
49   int GetDuty() { return duty; }
50   void SetDuty(int value);
51   void SetIntensity();
52
53   Run GetRun() { return run; }
54   void SetRun(Run value);
55
56   Polarity GetPolarity() { return polarity; }
57   void SetPolarity(Polarity value);
58
59   PWM(Pin pin);

```

```

52     ~PWM();
53
54     private:
55     int period;           // current period
56     int duty;            // current duty
57     float intensity;     // current intensity
58     uint8_t pixelvalue; // current pixelvalue
59     Run run;             // current run state
60     Polarity polarity; // current polarity
61
62     string basepath;    // the basepath ocp
63     string dutypath;    // base + duty path
64     string periodpath; // base + period path
65     string runpath;    // base + run path
66     string polaritypath; // base + polarity path
67
68     void calcIntensity();
69 };
70 }



---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
 * strictly prohibited
3   * and only allowed with the written consent of the author (
 * Jelle Spijker)
4   * This software is proprietary and confidential
5   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6   */
7
8 #include "PWM.h"
9
10 namespace Hardware {
11 /// <summary>
12 /// Constructeur
13 /// </summary>
14 /// <param name="pin">Pin</param>
15 PWM::PWM(Pin pin) {
16     this->pin = pin;
17
18     // Check if PWM cape is loaded, if not load it
19     if (!CapeLoaded(PWM_CAPE)) {
20         Write(SLOTS, PWM_CAPE);
21     }
22
23     // Init the pin
24     switch (pin) {
25         case Hardware::PWM::P8_13:
26             system("config-pin P8.13 pwm");
27             basepath = OCP_PATH;
28             basepath.append("pwmchip4/pwm1");
29             break;
30         case Hardware::PWM::P8_19:
31             system("config-pin P8.19 pwm");
32             basepath = OCP_PATH;
33             basepath.append("pwmchip4/pwm0");
34             break;

```

```
35     case Hardware::PWM::P9_14:
36         system("config-pin P9.14 pwm");
37         basepath = OCP_PATH;
38         basepath.append("pwmchip2/pwm0");
39         break;
40     case Hardware::PWM::P9_16:
41         system("config-pin P9.16 pwm");
42         basepath.append("pwmchip2/pwm1");
43         break;
44     }
45
46     // Get the working paths
47     dutypath = basepath + "/duty_cycle";
48     periodpath = basepath + "/period";
49     runpath = basepath + "/run";
50     polaritypath = basepath + "/polarity";
51
52     // Give Linux time to setup directory structure;
53     usleep(250000);
54
55     // Read current values
56     period = StringToNumber<int>(Read(periodpath));
57     duty = StringToNumber<int>(Read(dutypath));
58     run = static_cast<Run>(StringToNumber<int>(Read(runpath)))
59     ;
60     polarity = static_cast<Polarity>(StringToNumber<int>(Read(
61         polaritypath)));
62
63     // calculate the current intensity
64     calcIntensity();
65 }
66
67 PWM::~PWM() {}
68
69 /// <summary>
70 /// Calculate the current intensity
71 /// </summary>
72 void PWM::calcIntensity() {
73     if (polarity == Normal) {
74         if (duty == 0) {
75             intensity = 0.0f;
76         } else {
77             intensity = (float)period / (float)duty;
78         }
79     } else {
80         if (period == 0) {
81             intensity = 0.0f;
82         } else {
83             intensity = (float)duty / (float)period;
84         }
85     }
86 /// <summary>
87 /// Set the intensity level as percentage
88 /// </summary>
```

```
89  /// <param name="value">floating value multiplication factor
90  </param>
91  void PWM::SetIntensity(float value) {
92  if (polarity == Normal) {
93      SetDuty(static_cast<int>((value * duty) + 0.5));
94  } else {
95      SetPeriod(static_cast<int>((value * period) + 0.5));
96  }
97
98  /// <summary>
99  /// Set the output as a corresponding uint8_t value
100 /// </summary>
101 /// <param name="value">pixel value 0-255</param>
102 void PWM::SetPixelValue(uint8_t value) {
103 if (period != 255) {
104     SetPeriod(255);
105 }
106 SetDuty(255 - value);
107 pixelvalue = value;
108 }
109
110 /// <summary>
111 /// Set the period of the signal
112 /// </summary>
113 /// <param name="value">period : int</param>
114 void PWM::SetPeriod(int value) {
115     string valstr = NumberToString<int>(value);
116     Write(periodpath, valstr);
117     period = value;
118
119     calcIntensity();
120 }
121
122 /// <summary>
123 /// Set the duty of the signal
124 /// </summary>
125 /// <param name="value">duty : int</param>
126 void PWM::SetDuty(int value) {
127     string valstr = NumberToString<int>(value);
128     Write(dutypath, valstr);
129     duty = value;
130
131     calcIntensity();
132 }
133
134 /// <summary>
135 /// Run the signal
136 /// </summary>
137 /// <param name="value">On or Off</param>
138 void PWM::SetRun(Run value) {
139     int valInt = static_cast<int>(value);
140     string valstr = NumberToString<int>(valInt);
141     Write(runpath, valstr);
142     run = value;
143 }
```

```
144
145  /// <summary>
146  /// Set the polarity
147  /// </summary>
148  /// <param name="value">Normal or Inverted signal</param>
149  void PWM::SetPolarity(Polarity value) {
150      int valInt = static_cast<int>(value);
151      string valstr = NumberToString<int>(valInt);
152      Write(runpath, valstr);
153      polarity = value;
154  }
155 }
```

G.0.10 ADC Class

```

35 #define ADC6_PATH
36     \
37     "/sys/bus/iio/devices/iio:device0/in_voltage6_raw" /*!<
38     path to analogue pin \
39     \
39     "/sys/bus/iio/devices/iio:device0/in_voltage7_raw" /*!<
40     path to analogue pin \
41     \
42 namespace Hardware {
43 class ADC : public BBB {
44 public:
45     /*! Enumerator to indicate the analogue pin*/
46     enum ADCPin {
47         ADC0, /*!< AIN0 pin*/
48         ADC1, /*!< AIN1 pin*/
49         ADC2, /*!< AIN2 pin*/
50         ADC3, /*!< AIN3 pin*/
51         ADC4, /*!< AIN4 pin*/
52         ADC5, /*!< AIN5 pin*/
53         ADC6, /*!< AIN6 pin*/
54         ADC7 /*!< AIN7 pin*/
55     };
56
57     ADCPin Pin; /*!< current pin*/
58
59     ADC(APCPin pin);
60     ~ADC();
61
62     int GetCurrentValue();
63     float GetIntensity() { return Intensity; }
64     int GetMinIntensity() { return MinIntensity; }
65     int GetMaxIntensity() { return MaxIntensity; }
66
67     void SetMinIntensity();
68     void SetMaxIntensity();
69
70     int WaitForValueChange();
71     int WaitForValueChange(CallbackType callback);
72     void WaitForValueChangeCancel() { this->threadRunning =
73         false; }
74
74 private:
75     string adcpath; /*!< Path to analogue write file*/
76     float Intensity; /*!< Current intensity expressed as
77     percentage*/
77     int MinIntensity; /*!< Voltage level which represent 0
78     percentage*/
78     int MaxIntensity; /*!< Voltage level which represent 100
79     percentage*/

```

```
80     friend void *threadedPollADC(void *value); /*!< friend
81     polling function*/
82 }
83 void *threadedPollADC(void *value);
84 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
10 #include "ADC.h"
11
12 namespace Hardware {
13 /*! Constructor
14 \param pin and ADCPin type indicating which analogue pin to
15  use
16 */
17 ADC::ADC(ADCPin pin) {
18     this->Pin = pin;
19     switch (pin) {
20     case Hardware::ADC::ADCO:
21         adcpath = ADC0_PATH;
22         break;
23     case Hardware::ADC::ADC1:
24         adcpath = ADC1_PATH;
25         break;
26     case Hardware::ADC::ADC2:
27         adcpath = ADC2_PATH;
28         break;
29     case Hardware::ADC::ADC3:
30         adcpath = ADC3_PATH;
31         break;
32     case Hardware::ADC::ADC4:
33         adcpath = ADC4_PATH;
34         break;
35     case Hardware::ADC::ADC5:
36         adcpath = ADC5_PATH;
37         break;
38     case Hardware::ADC::ADC6:
39         adcpath = ADC6_PATH;
40         break;
41     case Hardware::ADC::ADC7:
42         adcpath = ADC7_PATH;
43         break;
44     }
45 }
46
47 MinIntensity = 0;
48 MaxIntensity = 4096;
49 }
```

```

47  /*! De-constructor*/
48  ADC::~ADC() {}
49
50  /*! Reads the current voltage in the pin
51  \return an integer between 0 and 4096
52  */
53  int ADC::GetCurrentValue() {
54      int retVal = StringToNumber<int>(Read(adcpPath));
55      Intensity = (float)(retVal - MinIntensity) /
56                      (4096 - (MinIntensity + (4096 - MaxIntensity)))
57                      );
58      return retVal;
59  }
60  /*! Set the current voltage at the pin as the minimum
61  voltage*/
62  void ADC::SetMinIntensity() {
63      MinIntensity = StringToNumber<int>(Read(adcpPath));
64  }
65  void ADC::SetMaxIntensity() {
66      MaxIntensity = StringToNumber<int>(Read(adcpPath));
67  }
68
69  /*! Threading enabled polling of the analogue pin
70  \param callback the function which should be called when
71  polling indicates a
72  change CallbackType
73  \return 0
74  */
75  int ADC::WaitForValueChange(CallbackType callback) {
76      threadRunning = true;
77      callbackFunction = callback;
78      if (pthread_create(&thread, NULL, &threadedPollADC,
79                          static_cast<void *>(this))) {
80          threadRunning = false;
81          throw Exception::
82              FailedToCreateGPIOPollingThreadException();
83      }
84      return 0;
85  }
86  /*! Polling of the analogue pin
87  \return the current value
88  */
89  int ADC::WaitForValueChange() {
90      int fd, i, epollfd, count = 0;
91      struct epoll_event ev;
92      epollfd = epoll_create(1);
93      if (epollfd == -1) {
94          throw Exception::
95              FailedToCreateGPIOPollingThreadException(
96                  "GPIO: Failed to create epollfd!");
97      }
98      if ((fd = open(adcpPath.c_str(), O_RDONLY | O_NONBLOCK)) ==
99          -1) {

```

```
97     throw Exception::ADCReadException();
98 }
99 ev.events = EPOLLIN;
100 ev.data.fd = fd;
101
102 if (epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev) == -1) {
103     throw Exception::
104         FailedToCreateGPIOPollingThreadException(
105             "ADC: Failed to add control interface!");
106 }
107 while (count <= 1) {
108     i = epoll_wait(epollfd, &ev, 1, -1);
109     if (i == -1) {
110         close(fd);
111         return -1;
112     } else {
113         count++;
114     }
115 }
116 close(fd);
117 return StringToNumber<int>(Read(adcpPath));
118 }
119
120 /*! friendly function to start the threading*/
121 void *threadedPollADC(void *value) {
122     ADC *adc = static_cast<ADC *>(value);
123     while (adc->threadRunning) {
124         adc->callbackFunction(adc->WaitForValueChange());
125         usleep(200000);
126     }
127 }
128 }
```

G.0.11 EC12P Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8   */
9
10 /*! \class EC12P
11 Interaction with the sparkfun RGB encoder
12 */
13
14 #pragma once
15
16 #include "eqep.h"
17 #include "GPIO.h"
18 #include "FailedToCreateThreadException.h"
19
20 #include <pthread.h>
21
22 using namespace std;
23
24 namespace Hardware {
25 class EC12P {
26 public:
27     EC12P();
28     ~EC12P();
29
30     /*! Enumerator indicating the color of the encoder shaft*/
31     enum Color {
32         Red,      /*!< Red*/
33         Pink,    /*!< Pink*/
34         Blue,    /*!< Blue*/
35         SkyBlue, /*!< SkyBlue*/
36         Green,   /*!< Green*/
37         Yellow,  /*!< Yellow*/
38         White,   /*!< White*/
39         None     /*!< Off*/
40     };
41
42     void SetPixelColor(Color value);
43     Color GetPixelColor() { return PixelColor; };
44
45     void RainbowLoop(int sleepperiod);
46     void StopRainbowLoop() { threadRunning = false; };
47
48     eQEP Rotary{eQEP2, eQEP::eQEP_Mode_Absolute}; /*!< The
49     encoder*/
50     GPIO Button{68};                                /*!< The
51     pushbutton*/
52
53 private:
54     Color PixelColor; /*!< Current shaft color*/
55 }
```

```

51
52     GPIO R{31}; /*!< Red LED*/
53     GPIO B{48}; /*!< Blue LED*/
54     GPIO G{51}; /*!< Green LED*/
55
56     pthread_t thread; /*!< the thread*/
57     bool threadRunning; /*!< Bool used to stop the thread*/
58     int sleepperiod; /*!< Sleep period*/
59     friend void *colorLoop(void *value);
60 };
61 void *colorLoop(void *value);
62 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
10 #include "EC12P.h"
11
12 namespace Hardware {
13 /*! Constructor*/
14 EC12P::EC12P() {
15     // Init Rotary button
16     Button.SetDirection(GPIO::Input);
17     Button.SetEdge(GPIO::Rising);
18
19     // Init Encoder
20     Rotary.set_period(100000000L);
21
22     // Init Encoder color
23     R.SetDirection(GPIO::Output);
24     B.SetDirection(GPIO::Output);
25     G.SetDirection(GPIO::Output);
26     SetPixelColor(None);
27 }
28
29 /*! De-constructor*/
30 EC12P::~EC12P() {}
31
32 /*! Set the shaft color
33 \param value as Color enumerator
34 */
35 void EC12P::SetPixelColor(Color value) {
36     switch (value) {
37     case Hardware::EC12P::Red:
38         R.SetValue(GPIO::High);
39         B.SetValue(GPIO::Low);
40         G.SetValue(GPIO::Low);
41         break;

```

```
42     case Hardware::EC12P::Pink:
43         R.SetValue(GPIO::High);
44         B.SetValue(GPIO::High);
45         G.SetValue(GPIO::Low);
46         break;
47     case Hardware::EC12P::Blue:
48         R.SetValue(GPIO::Low);
49         B.SetValue(GPIO::High);
50         G.SetValue(GPIO::Low);
51         break;
52     case Hardware::EC12P::SkyBlue:
53         R.SetValue(GPIO::Low);
54         B.SetValue(GPIO::High);
55         G.SetValue(GPIO::High);
56         break;
57     case Hardware::EC12P::Green:
58         R.SetValue(GPIO::Low);
59         B.SetValue(GPIO::Low);
60         G.SetValue(GPIO::High);
61         break;
62     case Hardware::EC12P::Yellow:
63         R.SetValue(GPIO::High);
64         B.SetValue(GPIO::Low);
65         G.SetValue(GPIO::High);
66         break;
67     case Hardware::EC12P::White:
68         R.SetValue(GPIO::High);
69         B.SetValue(GPIO::High);
70         G.SetValue(GPIO::High);
71         break;
72     case Hardware::EC12P::None:
73         R.SetValue(GPIO::Low);
74         B.SetValue(GPIO::Low);
75         G.SetValue(GPIO::Low);
76         break;
77     }
78     PixelColor = value;
79 }
80
81 /*! Loops through all the colors except of as a thread */
82 void EC12P::RainbowLoop(int sleepperiod) {
83     this->sleepperiod = sleepperiod;
84     this->threadRunning = true;
85     if (pthread_create(&thread, NULL, colorLoop, this)) {
86         throw Exception::FailedToCreateThreadException();
87     }
88 }
89
90 /*! The thread function that runs trough all the colors*/
91 void *colorLoop(void *value) {
92     int i = 0;
93     EC12P *ec12p = static_cast<EC12P *>(value);
94     EC12P::Color pcolor;
95     while (ec12p->threadRunning) {
96         pcolor = static_cast<EC12P::Color>(i);
97         ec12p->SetPixelColor(pcolor);
```

```
98     usleep(ec12p->sleepperiod);
99     i++;
100    if (i == 6) {
101        i = 0;
102    }
103 }
104 return ec12p;
105 }
106 }
```

G.0.12 eQep Class

```
1  /*
2  *  TI eQEP driver interface API
3  *
4  *  Copyright (C) 2013 Nathaniel R. Lewis - http://
5  *          nathanielrlewis.com/
6  *
7  *  This program is free software; you can redistribute it and
8  *  /or modify
9  *  it under the terms of the GNU General Public License as
10 *  published by
11 *  the Free Software Foundation; either version 2 of the
12 *  License, or
13 *  (at your option) any later version.
14 *
15 *  This program is distributed in the hope that it will be
16 *  useful,
17 *  but WITHOUT ANY WARRANTY; without even the implied
18 *  warranty of
19 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
20 *  the
21 *  GNU General Public License for more details.
22 *
23 *  You should have received a copy of the GNU General Public
24 *  License
25 *  along with this program; if not, write to the Free
26 *  Software
27 *  Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
28 *
29 *
30 *  This code is changed by Jelle Spijker (C) 2014.
31 *  Introducing polling with threading.
32 *
33 */
34
35 #pragma once
36
37 #include <iostream>
38 #include <stdint.h>
39 #include <string>
40 #include "BBB.h"
41
42 #define eQEPO "/sys/devices/ocp.3/48300000.epwmss/48300180.
43         eqep"
44 #define eQEP1 "/sys/devices/ocp.3/48302000.epwmss/48302180.
45         eqep"
46 #define eQEP2 "/sys/devices/ocp.3/48304000.epwmss/48304180.
47         eqep"
48
49 namespace Hardware {
50 // Class which defines an interface to my eQEP driver
51 class eQEP : public BBB {
52     // Base path for the eQEP unit
53     std::string path;
```

```

43 public:
44     // Modes of operation for the eQEP hardware
45     typedef enum {
46         // Absolute positioning mode
47         eQEP_Mode_Absolute = 0,
48
49         // Relative positioning mode
50         eQEP_Mode_Relative = 1,
51
52         // Error flag
53         eQEP_Mode_Error = 2,
54     } eQEP_Mode;
55
56     // Default constructor for the eQEP interface driver
57     eQEP(std::string _path, eQEP_Mode _mode);
58
59     // Reset the value of the encoder
60     void set_position(int32_t position);
61
62     // Get the position of the encoder, pass poll as true to
63     // poll the pin, whereas
64     // passing false reads the immediate value
65     int32_t get_position(bool _poll = true);
66
67     // Thread of the poll
68     int WaitForPositionChange(CallbackType callback);
69     void WaitForPositionChangeCancel() { this->threadRunning =
70         false; }
71
72     // Set the polling period
73     void set_period(long long unsigned int period);
74
75     // Get the polling period of the encoder
76     uint64_t get_period();
77
78     // Set the mode of the eQEP hardware
79     void set_mode(eQEP_Mode mode);
80
81     // Get the mode of the eQEP hardware
82     eQEP_Mode get_mode();
83
84     private:
85         friend void *threadedPolleqep(void *value);
86     };
87 }



---


1  /*
2  * TI eQEP driver interface API
3  *
4  * Copyright (C) 2013 Nathaniel R. Lewis - http://
5  * nathanielrlewis.com/
6  * This program is free software; you can redistribute it and
7  * /or modify

```

```
7 * it under the terms of the GNU General Public License as
8 * published by
9 * the Free Software Foundation; either version 2 of the
10 * License, or
11 * (at your option) any later version.
12 *
13 * This program is distributed in the hope that it will be
14 * useful,
15 * but WITHOUT ANY WARRANTY; without even the implied
16 * warranty of
17 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
18 * the
19 * GNU General Public License for more details.
20 *
21 * You should have received a copy of the GNU General Public
22 * License
23 * along with this program; if not, write to the Free
24 * Software
25 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
26 *
27 * This file is modified by Jelle Spijker 2014
28 * Added polling and threading capabilties
29 *
30 */
31
32
33 // Pull in our eQEP driver definitions
34 #include "eqep.h"
35
36
37 // Language dependencies
38 #include <cstdint>
39 #include <cstdlib>
40 #include <cstdio>
41
42 // POSIX dependencies
43 #include <unistd.h>
44 #include <fcntl.h>
45 #include <poll.h>
46 #include <sys/types.h>
47 #include <sys/stat.h>
48
49
50 namespace Hardware {
51 // Constructor for eQEP driver interface object
52 eQEP::eQEP(std::string _path, eQEP::eQEP_Mode _mode) : path(
53     _path) {
54     if (_path == eQEP0) {
55         if (!CapeLoaded("bone_eqep0")) {
56             Write(SLOTS, "bone_eqep0");
57         }
58     } else if (_path == eQEP1) {
59         if (!CapeLoaded("bone_eqep1")) {
60             Write(SLOTS, "bone_eqep1");
61         }
62     } else if (_path == eQEP2) {
63         if (!CapeLoaded("bone_eqep2b")) {
64             Write(SLOTS, "bone_eqep2b");
65         }
66     }
67 }
```

```
55     }
56
57     // Set the mode of the hardware
58     this->set_mode(_mode);
59
60     // Reset the position
61     this->set_position(0);
62 }
63
64 // Set the position of the eQEP hardware
65 void eQEP::set_position(int32_t position) {
66     // Open the file representing the position
67     FILE *fp = fopen((this->path + "/position").c_str(), "w");
68
69     // Check that we opened the file correctly
70     if (fp == NULL) {
71         // Error, break out
72         std::cerr << "[eQEP " << this->path << "] Unable to open
73             position for write"
74             << std::endl;
75     }
76
77     // Write the desired value to the file
78     fprintf(fp, "%d\n", position);
79
80     // Commit changes
81     fclose(fp);
82 }
83
84 // Set the period of the eQEP hardware
85 void eQEP::set_period(long long unsigned int period) {
86     // Open the file representing the position
87     FILE *fp = fopen((this->path + "/period").c_str(), "w");
88
89     // Check that we opened the file correctly
90     if (fp == NULL) {
91         // Error, break out
92         std::cerr << "[eQEP " << this->path << "] Unable to open
93             period for write"
94             << std::endl;
95     }
96
97     // Write the desired value to the file
98     fprintf(fp, "%llu\n", period);
99
100    // Commit changes
101    fclose(fp);
102 }
103
104 // Set the mode of the eQEP hardware
105 void eQEP::set_mode(eQEP::eQEP_Mode _mode) {
106     // Open the file representing the position
107     FILE *fp = fopen((this->path + "/mode").c_str(), "w");
108 }
```

```
109     // Check that we opened the file correctly
110     if (fp == NULL) {
111         // Error, break out
112         std::cerr << "[eQEP " << this->path << "] Unable to open
113             mode for write"
114             << std::endl;
115         return;
116     }
117
118     // Write the desired value to the file
119     fprintf(fp, "%u\n", _mode);
120
121     // Commit changes
122     fclose(fp);
123 }
124
125 int eQEP::WaitForPositionChange(CallbackType callback) {
126     threadRunning = true;
127     callbackFunction = callback;
128     if (pthread_create(&this->thread, NULL, &threadedPolleqep,
129                         static_cast<void *>(this))) {
130         threadRunning = false;
131         throw Exception::
132             FailedToCreateGPIOPollingThreadException();
133     }
134 }
135
136 // Get the position of the hardware
137 int32_t eQEP::get_position(bool _poll) {
138     // Position temporary variable
139     int32_t position;
140     char dummy;
141     struct pollfd ufd;
142
143     // Do we want to poll?
144     if (_poll) {
145         // Open a connection to the attribute file.
146         if ((ufd.fd = open((this->path + "/position").c_str(),
147                           O_RDWR)) < 0) {
148             // Error, break out
149             std::cerr << "[eQEP " << this->path
150                 << "] unable to open position for polling"
151                 << std::endl;
152         }
153
154         // Dummy read
155         read(ufd.fd, &dummy, 1);
156
157         // Poll the port
158         ufd.events = (short)EPOLLET;
159         if (poll(&ufd, 1, -1) < 0) {
160             // Error, break out
161             std::cerr << "[eQEP " << this->path << "] Error
```

```
161         occurred whilst polling"
162         << std::endl;
163         close(ufd.fd);
164     }
165 }
166
167 // Read the position
168 FILE *fp = fopen((this->path + "/position").c_str(), "r");
169
170 // Check that we opened the file correctly
171 if (fp == NULL) {
172     // Error, break out
173     std::cerr << "[eQEP " << this->path << "] Unable to open
174         position for read"
175         << std::endl;
176     close(ufd.fd);
177     return 0;
178 }
179
180 // Write the desired value to the file
181 fscanf(fp, "%d", &position);
182
183 // Commit changes
184 fclose(fp);
185
186 // If we were polling, close the polling file
187 if (_poll) {
188     close(ufd.fd);
189 }
190
191 // Return the position
192 return position;
193
194 // Get the period of the eQEP hardware
195 uint64_t eQEP::get_period() {
196     // Open the file representing the position
197     FILE *fp = fopen((this->path + "/period").c_str(), "r");
198
199     // Check that we opened the file correctly
200     if (fp == NULL) {
201         // Error, break out
202         std::cerr << "[eQEP " << this->path << "] Unable to open
203             period for read"
204             << std::endl;
205         return 0;
206     }
207
208     // Write the desired value to the file
209     uint64_t period = 0;
210     fscanf(fp, "%llu", &period);
211
212     // Commit changes
213     fclose(fp);
214 }
```

```
214     // Return the period
215     return period;
216 }
217
218 // Get the mode of the eQEP hardware
219 eQEP::eQEP_Mode eQEP::get_mode() {
220     // Open the file representing the position
221     FILE *fp = fopen((this->path + "/mode").c_str(), "r");
222
223     // Check that we opened the file correctly
224     if (fp == NULL) {
225         // Error, break out
226         std::cerr << "[eQEP " << this->path << "] Unable to open
227             mode for read"
228             << std::endl;
229         return eQEP::eQEP_Mode_Error;
230     }
231
232     // Write the desired value to the file
233     eQEP::eQEP_Mode mode;
234     fscanf(fp, "%u", (unsigned int *)&mode);
235
236     // Commit changes
237     fclose(fp);
238
239     // Return the mode
240     return mode;
241 }
242
243 void *threadedPolleqep(void *value) {
244     eQEP *eqep = static_cast<eQEP *>(value);
245     while (eqep->threadRunning) {
246         eqep->callbackFunction(eqep->get_position(true));
247         usleep(eqep->debounceTime * 1000);
248     }
249 }
250 }
```

G.0.13 SoilCape Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkerspijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include "EC12P.h"
13 #include "GPIO.h"
14 #include "PWM.h"
15 #include "ADC.h"
16
17 namespace Hardware {
18 class SoilCape {
19 public:
20     EC12P RGBEncoder;
21     PWM MicroscopeLEDs{PWM::P9_14};
22     ADC MicroscopeLDR{ADC::ADC0};
23
24     SoilCape();
25     ~SoilCape();
26 };
27 }
```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkerspijker.jelle@gmail.com>, 2015
8   */
9
10 #include "SoilCape.h"
11
12 namespace Hardware {
13     SoilCape::SoilCape() {}
14     SoilCape::~SoilCape() {}
15 }
```

G.0.14 USB Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include <stdio.h>
13 #include <unistd.h>
14 #include <fcntl.h>
15 #include <errno.h>
16 #include <sys/ioctl.h>
17
18 namespace Hardware {
19 class USB {
20 public:
21     USB();
22     ~USB();
23     void ResetUSB();
24 };
25 }
```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10 #include "USB.h"
11
12 namespace Hardware {
13 USB::USB() {}
14
15 USB::~USB() {}
16
17 void USB::ResetUSB() {
18     int fd, rc;
19
20     fd = open("/dev/bus/usb/001/002", O_WRONLY);
21     rc = ioctl(fd, USBDEVFS_RESET, 0);
22     if (rc < 0) {
23         throw - 1;
24     }
25     close(fd);
26 }
```


G.0.15 General project files

```
1 #-----
2 #
3 # Project created by QtCreator 2015-06-06T10:49:23
4 #
5 #-----
6
7 QT      -= core gui
8
9 TARGET = SoilHardware
10 TEMPLATE = lib
11 VERSION = 0.9.1
12
13 DEFINES += SOILHARDWARE_LIBRARY
14 QMAKE_CXXFLAGS += -std=c++11 -pthread
15 unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../../../build/install/
16
17 SOURCES += \
18     USB.cpp \
19     SoilCape.cpp \
20     PWM.cpp \
21     Microscope.cpp \
22     GPIO.cpp \
23     eqep.cpp \
24     EC12P.cpp \
25     BBB.cpp \
26     ADC.cpp
27
28 HEADERS += \
29     ValueOutOfBoundsException.h \
30     USB.h \
31     SoilCape.h \
32     PWM.h \
33     MicroscopeNotFoundException.h \
34     Microscope.h \
35     Hardware.h \
36     GPIOReadException.h \
37     GPIO.h \
38     FailedToCreateThreadException.h \
39     FailedToCreateGPIOPollingThreadException.h \
40     eqep.h \
41     EC12P.h \
42     CouldNotGrabImageException.h \
43     BBB.h \
44     ADCReadException.h \
45     ADC.h
46
47 #opencv
48 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui - \
49         -lopencv_photo -lopencv_imgcodecs -lopencv_videoio
50 INCLUDEPATH += /usr/local/include/opencv
51 INCLUDEPATH += /usr/local/include
52 #boost
53 DEFINES += BOOST_ALL_DYN_LINK
```

```
54 INCLUDEPATH += /usr/include/boost
55 LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_filesystem -
      -lboost_system
56
57 unix {
58     target.path = $PWD/../../../../build/install
59     INSTALLS += target
60 }


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11
12 #include "ADC.h"
13 #include "EC12P.h"
14 #include "eqep.h"
15 #include "GPIO.h"
16 #include "PWM.h"
17 #include "SoilCape.h"
18 #include "Microscope.h"
19 #include "CouldNotGrabImageException.h"
20 #include "ADCReadException.h"
21 #include "FailedToCreateGPIOPollingThreadException.h"
22 #include "FailedToCreateThreadException.h"
23 #include "GPIOReadException.h"
24 #include "MicroscopeNotFoundException.h"
25 #include "ValueOutOfBoundsException.h"


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8 */
9
10 #include <exception>
11 #include <string>
12
13 using namespace std;
14
15 namespace Hardware {
16 namespace Exception {
17 class ValueOutOfBoundsException : public std::exception {
18 public:
```

```
19  ValueOutOfBoundsException(string m = "Value out of bounds!
20  " ) : msg(m){};
21  ~ValueOutOfBoundsException() _GLIBCXX_USE_NOEXCEPT{};
22  const char *what() const _GLIBCXX_USE_NOEXCEPT { return
23  msg.c_str(); };
24
25  private:
26  string msg;
27 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11 #include <exception>
12 #include <string>
13
14 using namespace std;
15
16 namespace Hardware {
17 namespace Exception {
18 class ADCReadException : public std::exception {
19 public:
20  ADCReadException(string m = "Can't read ADC data!") : msg(
21  m){};
22  ~ADCReadException() _GLIBCXX_USE_NOEXCEPT{};
23  const char *what() const _GLIBCXX_USE_NOEXCEPT { return
24  msg.c_str(); };
25
26 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
10 #include <exception>
11 #include <string>
```

```

12
13  using namespace std;
14
15  namespace Hardware {
16  namespace Exception {
17  class FailedToCreateGPIOPollingThreadException : public std
18    ::exception {
18  public:
19    FailedToCreateGPIOPollingThreadException(
20      string m = "Failed to create GPIO polling thread!")
21      : msg(m){};
22    ~FailedToCreateGPIOPollingThreadException() _GLIBCXX_USE_NOEXCEPT{};
23    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
24      msg.c_str(); };
25
25  private:
26    string msg;
27  };
28 }
29 }



---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
8 #pragma once
10
10 #include <exception>
11 #include <string>
12
13  using namespace std;
14
15  namespace Hardware {
16  namespace Exception {
17  class FailedToCreateThreadException : public std::exception
18    {
18  public:
19    FailedToCreateThreadException(string m = "Couldn't create
20      the thread!")
21      : msg(m){};
22    ~FailedToCreateThreadException() _GLIBCXX_USE_NOEXCEPT{};
23    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
24      msg.c_str(); };
25
25  private:
26    string msg;
27  };
28 }
28

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10 #define EXCEPTION_OPENCAM "Exception could not open cam!"
11 #define EXCEPTION_OPENCAM_NR 0
12 #define EXCEPTION_NOCAMS "Exception no cam available!"
13 #define EXCEPTION_NOCAMS_NR 1
14 #define EXCEPTION_QUERY "Exception could not query device!"
15 #define EXCEPTION_QUERY_NR 3
16 #define EXCEPTION_FORMAT_RESOLUTION "Exception No supported
17   formats and resolutions!"
18 #define EXCEPTION_FORMAT_RESOLUTION_NR 4
19 #define EXCEPTION_CTRL_NOT_FOUND "Control not found!"
20 #define EXCEPTION_CTRL_NOT_FOUND_NR 5
21 #define EXCEPTION_CTRL_VALUE "Control value not set!"
22 #define EXCEPTION_CTRL_VALUE_NR 5
23
24
25 #pragma once
26 #include <exception>
27 #include <string>
28
29 using namespace std;
30
31 namespace Hardware {
32 namespace Exception {
33 class MicroscopeException : public std::exception {
34 public:
35     MicroscopeException(string m = EXCEPTION_OPENCAM,
36                         int n = EXCEPTION_OPENCAM_NR) : msg{m
37                         }, nr{n} { }
38     ~MicroscopeException() __GLIBCXX_USE_NOEXCEPT {}
39     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
40         msg.c_str(); }
41     const int *id() const __GLIBCXX_USE_NOEXCEPT { return &nr;
42     }
43
44 private:
45     string msg;
46     int nr;
47 };
48 }
49

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential

```

```

5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6  */
7
8 #pragma once
9 #include <exception>
10 #include <string>
11
12 using namespace std;
13
14 namespace Hardware {
15 namespace Exception {
16 class CouldNotGrabImageException : public std::exception {
17 public:
18     CouldNotGrabImageException(string m = "Unable to grab the
19         next image!")
20         : msg(m){};
21     ~CouldNotGrabImageException() __GLIBCXX_USE_NOEXCEPT{};
22     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
23         msg.c_str(); };
24
25 private:
26     string msg;
27 };

```

```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11
12 #include <exception>
13 #include <string>
14
15 using namespace std;
16
17 namespace Hardware {
18 namespace Exception {
19 class GPIOReadException : public std::exception {
20 public:
21     GPIOReadException(string m = "Can't read GPIO data!") :
22         msg(m){};
23     ~GPIOReadException() __GLIBCXX_USE_NOEXCEPT{};
24     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
25         msg.c_str(); };
26
27 private:
28     string msg;
29 };

```

```
27 }



---



```
1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2 * Unauthorized copying of this file, via any medium is
3 * strictly prohibited
4 * and only allowed with the written consent of the author (
5 * Jelle Spijker)
6 * This software is proprietary and confidential
7 * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11
12
13 using namespace std;
14
15 namespace Hardware {
16 namespace Exception {
17 class GPIOReadException : public std::exception {
18 public:
19 GPIOReadException(string m = "Can't read GPIO data!") :
20 msg(m){};
21 ~GPIOReadException() __GLIBCXX_USE_NOEXCEPT{};
22 const char *what() const __GLIBCXX_USE_NOEXCEPT { return
23 msg.c_str(); };
24
25 private:
26 string msg;
27 }
```



---


```



H. Vision Library

H.0.16 Image processing Class

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11 /*! Current class version*/
12 #define IMAGEPROCESSING_VERSION 1
13
14 /*! MACRO which sets the original pointer to the original
15   * image or a clone of
16   * the earlier processed image */
17 #define CHAIN_PROCESS(chain, 0, type)
18
19 if (chain) {
20
21     \
22     TempImg = ProcessedImg.clone();
23
24     0 = (type *)TempImg.data;
25
26 } else {
27
28     \
29     0 = (type *)OriginalImg.data;
30 }
```

```
21 /*! MACRO which trows an EmtpyImageException if the matrix
22    is empty*/
23 #define EMPTY_CHECK(img)
24
25     if (img.empty()) {
26
27         \
28         throw Exception::EmtpyImageException();
29     \
30
31 }
32
33 #include <opencv2/core.hpp>
34 #include <opencv2/highgui.hpp>
35 #include <opencv2/imgproc.hpp>
36
37 #include <stdint.h>
38 #include <cmath>
39 #include <vector>
40 #include <string>
41
42 #include <boost/signals2.hpp>
43 #include <boost/bind.hpp>
44
45 #include "EmptyImageException.h"
46 #include "WrongKernelSizeException.h"
47 #include "ChannelMismatchException.h"
48 #include "PixelValueOutOfBoundException.h"
49 #include "VisionDebug.h"
50
51 using namespace cv;
52
53 namespace Vision {
54 class ImageProcessing {
55 public:
56     typedef boost::signals2::signal<void(float, std::string)>
57     Progress_t;
58     boost::signals2::connection
59     connect_Progress(const Progress_t::slot_type &subscriber);
60
61 protected:
62     uchar *GetNRow(int nData, int hKsize, int nCols, uint32_t
63     totalRows);
64     Mat TempImg;
65
66     Progress_t prog_sig;
67
68 public:
69     ImageProcessing();
70     ~ImageProcessing();
71     Mat OriginalImg;
72     Mat ProcessedImg;
73
74     static void getOriententated(Mat &BW, cv::Point_<double> &
75         centroid,
76
77         double &theta, double &
78         eccentricity);
```

```

68     static void RotateImg(Mat &src, Mat &dst, double &theta,
69         cv::Point_<double> &Centroid, Rect &ROI);
70
71     double currentProg = 0.;
72     double ProgStep = 0.;
73
74     static std::vector<Mat> extractChannel(const Mat &src);
75
76     /*! Copy a matrix to a new matrix with a LUT mask
77     \param src the source image
78     \param *LUT type T with a LUT to filter out unwanted pixel
79     values
80     \param cvType an in where you can pas CV_UC8C1 etc.
81     \return The new matrix
82     */
83     template <typename T1, typename T2>
84     static Mat CopyMat(const Mat &src, T1 *LUT, int cvType) {
85         Mat dst(src.size(), cvType);
86         uint32_t nData = src.rows * src.cols * dst.step[1];
87         if (cvType == 0 || cvType == 8 || cvType == 16 || cvType
88             == 24) {
89             for (uint32_t i = 0; i < nData; i += dst.step[1]) {
90                 dst.data[i] =
91                     static_cast<uint8_t>(LUT[*(T2 *) (src.data + (i *
92                         src.step[1]))]);
93             }
94         } else if (cvType == 1 || cvType == 9 || cvType == 17 ||

95             cvType == 25) {
96             for (uint32_t i = 0; i < nData; i += src.step[1]) {
97                 dst.data[i] =
98                     static_cast<int8_t>(LUT[*(T2 *) (src.data + (i *
99                         src.step[1]))]);
100            }
101        } else if (cvType == 2 || cvType == 10 || cvType == 18
102             || cvType == 26) {
103            for (uint32_t i = 0; i < nData; i += src.step[1]) {
104                dst.data[i] =
105                    static_cast<uint16_t>(LUT[*(T2 *) (src.data + (i *
106                         src.step[1]))]);
107            }
108        } else if (cvType == 4 || cvType == 12 || cvType == 20
109             || cvType == 28) {
110            for (uint32_t i = 0; i < nData; i += src.step[1]) {
111                dst.data[i] =
112                    static_cast<int32_t>(LUT[*(T2 *) (src.data + (i *
113                         src.step[1]))]);
114            }
115        }
116    }
117
118    return dst;

```

```

112     }
113
114     /*! Copy a matrix to a new matrix with a mask
115     \param src the source image
116     \param *LUT type T with a LUT to filter out unwanted pixel
117     values
118     \param cvType an in where you can pas CV_UC8C1 etc.
119     \return The new matrix
120 */
121 template <typename T1>
122 static Mat CopyMat(const Mat &src, const Mat &mask, int
123     cvType) {
124     if (src.size != mask.size) {
125         throw Exception::WrongKernelSizeException(
126             "Mask not the same size as src Exception!");
127     }
128     if (mask.channels() != 1) {
129         throw Exception::WrongKernelSizeException(
130             "Mask has more then 1 channel Exception!");
131     }
132     Mat dst(src.size(), cvType);
133
134     vector<Mat> exSrc = Vision::ImageProcessing::
135         extractChannel(src);
136     vector<Mat> exDst;
137
138     int cvBaseType = cvType % 8;
139     for_each(exSrc.begin(), exSrc.end(), [&](&b>const Mat &
140         sItem) {
141         Mat dItem(src.size(), cvBaseType);
142         std::transform(sItem.begin<T1>(), sItem.end<T1>(),
143             mask.begin<T1>(),
144                 dItem.begin<T1>(),
145                 [](&b>const T1 &s, const T1 &m) -> T1 {
146                     return s * m; });
147         exDst.push_back(dItem);
148     });
149     merge(exDst, dst);
150
151     return dst;
152 }
153
154
155 template <typename T1>
156 static void ShowDebugImg(cv::Mat img, T1 maxVal, std::
157     string windowName,
158                 bool scale = true) {
159     if (img.rows > 0 && img.cols > 0) {
160         cv::Mat tempImg(img.size(), img.type());

```

```

160     if (scale == true) {
161         std::vector<cv::Mat> exSrc = extractChannel(img);
162         std::vector<cv::Mat> exDst;
163         int cvBaseType = img.type() % 8;
164         T1 MatMin = std::numeric_limits<T1>::max();
165         T1 MatMax = std::numeric_limits<T1>::min();
166
167         // Find the global max and min
168         for_each(exSrc.begin(), exSrc.end(), [&](const Mat &
169             sItem) {
170             std::for_each(sItem.begin<T1>(), sItem.end<T1>(),
171                         [&](const T1 &s) {
172                 if (s > MatMax) {
173                     MatMax = s;
174                 } else if (s < MatMin) {
175                     MatMin = s;
176                 }
177             });
178             int Range = MatMax - MatMin;
179             if (Range < 1)
180                 Range = maxVal;
181
182             // Convert the values
183             for_each(exSrc.begin(), exSrc.end(), [&](const cv::
184                 Mat &sItem) {
185                 Mat dItem(img.size(), cvBaseType);
186                 std::transform(sItem.begin<T1>(), sItem.end<T1>(),
187                               dItem.begin<T1>(),
188                               [&](const T1 &s) -> T1 {
189                                 return (T1)round(((s - MatMin) *
190                                     maxVal) / Range);
191                             });
192                 exDst.push_back(dItem);
193             });
194
195             merge(exDst, tempImg);
196         } else {
197             tempImg = img;
198         }
199         cv::namedWindow(windowName, cv::WINDOW_NORMAL);
200         cv::imshow(windowName, tempImg);
201         cv::waitKey(0);
202         cv::destroyWindow(windowName);
203     };
204 };
205 };
206 };

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential

```

```

5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6  */
7
8  /*! \class ImageProcessing
9  \brief Core class of all the image classes
10 Core class of all the image classes with a few commonly
    shared functions and
11 variables
12 */
13 #include "ImageProcessing.h"
14
15 namespace Vision {
16 /*! Constructor of the core class*/
17 ImageProcessing::ImageProcessing() {}
18
19 /*! De-constructor of the core class*/
20 ImageProcessing::~ImageProcessing() {}
21
22 /*! Create a LUT indicating which iteration variable i is
    the end of an row
23 \param nData an int indicating total pixels
24 \param hKsize int half the size of the kernel, if any. which
    acts as an offset
25 from the border pixels
26 \param nCols int number of columns in a row
27 \return array of uchar where a zero is a middle column and
    a 1 indicates an end
28 of an row minus the offset from half the kernel size
29 */
30 uchar *ImageProcessing::GetNRow(int nData, int hKsize, int
    nCols,
                               uint32_t totalRows) {
31     // Create LUT to determine when there is a new row
32     uchar *nRow = new uchar[nData + 1]{};
33     // int i = 0;
34     int shift = nCols - hKsize - 1;
35     for (uint32_t i = 0; i < totalRows; i++) {
36         nRow[(i * nCols) + shift] = 1;
37     }
38     return nRow;
39 }
40 }
41
42 std::vector<Mat> ImageProcessing::extractChannel(const Mat &
    src) {
43     vector<Mat> chans;
44     split(src, chans);
45     return chans;
46 }
47
48 void ImageProcessing::getOrientented(cv::Mat &BW, cv::Point_
    <double> &centroid,
49                                         double &theta, double &
    eccentricity) {
50     cv::Moments Mu = cv::moments(BW, true);
51     centroid.x = Mu.m10 / Mu.m00;

```

```

53     centroid.y = Mu.m01 / Mu.m00;
54
55     theta = 0;
56     double muPrime20 = (Mu.m20 / Mu.m00) - pow(centroid.x, 2);
57     double muPrime02 = (Mu.m02 / Mu.m00) - pow(centroid.y, 2);
58     double diffmuprime2 = muPrime20 - muPrime02;
59     double muPrime11 = (Mu.m11 / Mu.m00) - (centroid.x *
60                           centroid.y);
61
62     if (diffmuprime2 != 0) {
63         theta = 0.5 * atan((2 * muPrime11) / diffmuprime2);
64     }
65
66     double term1 = (muPrime20 + muPrime02) / 2;
67     double term2 = sqrt(4 * pow(muPrime11, 2) + pow(
68                           diffmuprime2, 2)) / 2;
69     eccentricity = sqrt(1 - (term1 - term2) / (term1 + term2));
70 }
71
72 void ImageProcessing::RotateImg(Mat &src, Mat &dst, double &
73                                 theta,
74                                 cv::Point_<double> &Centroid
75                                 , cv::Rect &ROI) {
76
77     cv::Mat temp;
78     temp.setTo(0);
79     double alpha = cos(theta);
80     double beta = sin(theta);
81     double cx = src.cols / 2;
82     double cy = src.rows / 2;
83     double dx = cx - Centroid.x;
84     double dy = cy - Centroid.y;
85     double rotData[2][3]{{alpha, beta, alpha * dx + beta * dy
86                           + Centroid.x},
87                           {-beta, alpha, alpha * dy + beta * dx
88                           + Centroid.y}};
89
90     cv::Mat totalrot(2, 3, CV_64FC1, rotData);
91
92     cv::warpAffine(src, temp, totalrot, cv::Size(src.rows *
93                                                 2.5, src.cols * 2.5),
94                                                 INTER_LINEAR);
95
96     // determine the actual ROI
97     cv::Point minP(0, 0);
98
99     if (src.channels() == 1) {
100         uchar *0 = temp.data;
101         uint32_t nData = temp.rows * temp.cols;
102         minP.x = temp.rows;
103         minP.y = temp.cols;
104         cv::Point maxP(0, 0);
105         int X, Y;
106         for (uint32_t i = 0; i < nData; i++) {
107             if (0[i] != 0) {
108                 Y = floor(i / temp.cols);
109                 X = (i % temp.cols);
110                 if (X < minP.x) {
111                     minP.x = X;
112                 }
113             }
114         }
115     }

```

```
102         if (Y < minP.y) {
103             minP.y = Y;
104         }
105         if (X > maxP.x) {
106             maxP.x = X;
107         }
108         if (Y > maxP.y) {
109             maxP.y = Y;
110         }
111     }
112 }
113 ROI = cv::Rect(minP, maxP);
114 }
115
116 if (src.channels() > 1) {
117     Centroid.x -= cx;
118     Centroid.y -= cy;
119
120     double xnew = Centroid.x * alpha - Centroid.y * beta;
121     double ynew = Centroid.x * beta - Centroid.y * alpha;
122
123     Centroid.x = xnew + cx + minP.x;
124     Centroid.y = ynew + cy + minP.y;
125 }
126 dst = temp(ROI).clone();
127 }
128
129 boost::signals2::connection
130 ImageProcessing::connect_Progress(const Progress_t::
131     slot_type &subscriber) {
132     return prog_sig.connect(subscriber);
133 }
```

H.0.17 Conversion Class

```

48                                     according to Matlab */
49 // float whitePoint[3] = { 0.9642, 1.0000, 0.8251 }; /*!<
50     Natural whitepoint
51 // in XYZ colorspace D50 according to Matlab */
52
52 void Lab2RI(float *0, float *P, int nData);
53 void RGB2XYZ(uchar *0, float *P, int nData);
54 void XYZ2Lab(float *0, float *P, int nData);
55 void RGB2Intensity(uchar *0, uchar *P, int nData);
56 inline float f_xyz2lab(float t);
57 };
58 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
8 /*! \class Conversion
9  class which converts a cv::Mat image from one colorspace to
10 the next colorspace
10 */
11 #include "Conversion.h"
12 namespace Vision {
13 /*! Constructor of the class */
14 Conversion::Conversion() {
15     OriginalColorSpace = None;
16     ProcessedColorSpace = None;
17 }
18
19 /*! Constructor of the class
20 \param src a cv::Mat object which is the source image
21 */
22 Conversion::Conversion(const Mat &src) {
23     OriginalColorSpace = None;
24     ProcessedColorSpace = None;
25     OriginalImg = src;
26 }
27
28 /*! Copy constructor*/
29 Conversion::Conversion(const Conversion &rhs) {
30     this->OriginalColorSpace = rhs.OriginalColorSpace;
31     this->OriginalImg = rhs.OriginalImg;
32     this->ProcessedColorSpace = rhs.ProcessedColorSpace;
33     this->ProcessedImg = rhs.ProcessedImg;
34     this->TempImg = rhs.TempImg;
35 }
36
37 /*! De-constructor of the class*/
38 Conversion::~Conversion() {}
39
40 /*! Assignment operator*/

```

```

41 Conversion &Conversion::operator=(Conversion rhs) {
42     if (&rhs != this) {
43         this->OriginalColorSpace = rhs.OriginalColorSpace;
44         this->OriginalImg = rhs.OriginalImg;
45         this->ProcessedColorSpace = rhs.ProcessedColorSpace;
46         this->ProcessedImg = rhs.ProcessedImg;
47         this->TempImg = rhs.TempImg;
48     }
49     return *this;
50 }
51
52 /*! Convert the source image from one colorspace to a
53    destination colorspace
54 - RGB 2 Intensity
55 - RGB 2 XYZ
56 - RGB 2 Lab
57 - RGB 2 Redness Index
58 - XYZ 2 Lab
59 - XYZ 2 Redness Index
60 - Lab 2 Redness Index
61 \param src a cv::Mat object which is the source image
62 \param dst a cv::Mat object which is the destination image
63 \param convertFrom the starting colorspace
64 \param convertTo the destination colorspace
65 \param chain use the results from the previous operation
66     default value = false;
67 */
68 void Conversion::Convert(const Mat &src, Mat &dst,
69                         ColorSpace convertFrom,
70                         ColorSpace convertTo, bool chain) {
71     OriginalImg = src;
72     Convert(convertFrom, convertTo, chain);
73     dst = ProcessedImg;
74 }
75
76 /*! Convert the source image from one colorspace to a
77    destination colorspace
78 possibilities are:
79 - RGB 2 Intensity
80 - RGB 2 XYZ
81 - RGB 2 Lab
82 - RGB 2 Redness Index
83 - XYZ 2 Lab
84 - XYZ 2 Redness Index
85 - Lab 2 Redness Index
86 \param convertFrom the starting colorspace
87 \param convertTo the destination colorspace
88 \param chain use the results from the previous operation
89     default value = false;
90 */
91 void Conversion::Convert(ColorSpace convertFrom, ColorSpace
92                         convertTo,
93                         bool chain) {
94     OriginalColorSpace = convertFrom;
95     ProcessedColorSpace = convertTo;
96 }
```

```

91 // Exception handling
92 EMPTY_CHECK(OriginalImg);
93 currentProg = 0.;
94 prog_sig(currentProg, "Converting colorspace");
95
96 int nData = OriginalImg.rows * OriginalImg.cols;
97 // uint32_t i, j;
98
99 if (convertFrom == RGB && convertTo == Intensity) // RGB 2
100   Intensity
101 {
102   ProcessedImg.create(OriginalImg.size(), CV_8UC1);
103   uchar *P = ProcessedImg.data;
104   uchar *0;
105   CHAIN_PROCESS(chain, 0, uchar);
106
107   prog_sig(currentProg, "RGB 2 Intensity conversion");
108   RGB2Intensity(0, P, nData);
109   currentProg += ProgStep;
110   prog_sig(currentProg, "RGB 2 Intensity conversion
111   Finished");
112 } else if (convertFrom == RGB && convertTo == CIE_XYZ) // RGB 2 XYZ
113 {
114   ProcessedImg.create(OriginalImg.size(), CV_32FC3);
115   float *P = (float *)ProcessedImg.data;
116   uchar *0;
117   CHAIN_PROCESS(chain, 0, uchar);
118
119   prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
120   RGB2XYZ(0, P, nData);
121   currentProg += ProgStep;
122   prog_sig(currentProg, "RGB 2 CIE XYZ conversion Finished
123   ");
124 } else if (convertFrom == RGB && convertTo == CIE_lab) // RGB 2 Lab
125 {
126   ProcessedImg.create(OriginalImg.size(), CV_32FC3);
127   float *P = (float *)ProcessedImg.data;
128   uchar *0;
129   CHAIN_PROCESS(chain, 0, uchar);
130
131   prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
132   RGB2XYZ(0, P, nData);
133   currentProg += ProgStep;
134   prog_sig(currentProg, "RGB 2 CIE XYZ conversion Finished
135   ");
136   Convert(CIE_XYZ, CIE_lab, true);
137 } else if (convertFrom == RGB && convertTo == RI) // RGB 2
138   RI
139 {
140   ProcessedImg.create(OriginalImg.size(), CV_32FC3);
141   float *P = (float *)ProcessedImg.data;
142   uchar *0;
143   CHAIN_PROCESS(chain, 0, uchar);
144 }
```

```

140     prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
141     RGB2XYZ(0, P, nData);
142     currentProg += ProgStep;
143     prog_sig(currentProg, "RGB 2 CIE XYZ conversion Finished
144     ");
144     Convert(CIE_XYZ, CIE_lab, true);
145     Convert(CIE_lab, RI, true);
146 } else if (convertFrom == CIE_XYZ && convertTo == CIE_lab)
147     // XYZ 2 Lab
148 {
149     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
150     float *P = (float *)ProcessedImg.data;
151     float *O;
152     CHAIN_PROCESS(chain, 0, float);
153
154     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion");
155     XYZ2Lab(0, P, nData);
156     currentProg += ProgStep;
157     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion
158     Finished");
159 } else if (convertFrom == CIE_XYZ && convertTo == RI) // // XYZ 2 RI
160 {
161     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
162     float *P = (float *)ProcessedImg.data;
163     float *O;
164     CHAIN_PROCESS(chain, 0, float);
165
166     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion");
167     XYZ2Lab(0, P, nData);
168     currentProg += ProgStep;
169     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion
170     Finished");
171     Convert(CIE_lab, RI, true);
172 } else if (convertFrom == CIE_lab && convertTo == RI) // // Lab 2 RI
173 {
174     ProcessedImg.create(OriginalImg.size(), CV_32FC1);
175     float *P = (float *)ProcessedImg.data;
176     float *O;
177     CHAIN_PROCESS(chain, 0, float);
178
179     prog_sig(currentProg, "CIE La*b* 2 Redness Index
180     conversion");
181     Lab2RI(0, P, nData * 3);
182     currentProg += ProgStep;
183     prog_sig(currentProg, "CIE La*b* 2 Redness Index
184     conversion Finsihed");
185 } else {
186     throw Exception::ConversionNotSupportedException();
187 }
188
189 /*! Conversion from RGB to Intensity
190 \param O a uchar pointer to the source image
191 \param P a uchar pointer to the destination image

```

```

188  \param nData an int indicating the total number of pixels
189  */
190  void Conversion::RGB2Intensity(uchar *O, uchar *P, int nData
191  ) {
192      uint32_t i;
193      int j;
194      i = 0;
195      j = 0;
196      while (j < nData) {
197          P[j++] = (*(O + i + 2) * 0.2126 + *(O + i + 1) * 0.7152
198          +
199          *(O + i) * 0.0722); // Grey value
200          i += 3;
201      }
202  }
203  /*! Conversion from RGB to CIE XYZ
204  \param O a uchar pointer to the source image
205  \param P a uchar pointer to the destination image
206  \param nData an int indicating the total number of pixels
207  */
208  void Conversion::RGB2XYZ(uchar *O, float *P, int nData) {
209      uint32_t endData = nData * OriginalImg.step.buf[1];
210      float R, G, B;
211      for (uint32_t i = 0; i < endData; i += OriginalImg.step.
212          buf[1]) {
213          R = static_cast<float>(*(O + i + 2) / 255.0f);
214          B = static_cast<float>(*(O + i + 1) / 255.0f);
215          G = static_cast<float>(*(O + i) / 255.0f);
216          P[i] = (XYZmat[0][0] * R) + (XYZmat[0][1] * B) + (XYZmat
217          [0][2] * G); // X
218          P[i + 1] = (XYZmat[1][0] * R) + (XYZmat[1][1] * B) + (
219          XYZmat[1][2] * G); // Y
220          P[i + 2] = (XYZmat[2][0] * R) + (XYZmat[2][1] * B) + (
221          XYZmat[2][2] * G); // Z
222      }
223  }
224  /*! Conversion from CIE XYZ to CIE La*b*
225  \param O a uchar pointer to the source image
226  \param P a uchar pointer to the destination image
227  \param nData an int indicating the total number of pixels
228  */
229  void Conversion::XYZ2Lab(float *O, float *P, int nData) {
230      uint32_t endData = nData * 3;
231      float yy0, xx0, zz0;
232      for (size_t i = 0; i < endData; i += 3) {
233          xx0 = *(O + i) / whitePoint[0];
234          yy0 = *(O + i + 1) / whitePoint[1];
235          zz0 = *(O + i + 2) / whitePoint[2];
236
237          if (yy0 > 0.008856) {
238              P[i] = (116 * pow(yy0, 0.333f)) - 16; // L
239          } else {
240              P[i] = 903.3 * yy0; // L
241          }
242      }
243  }

```

```
238
239     P[i + 1] = 500 * (f_xyz2lab(xx0) - f_xyz2lab(yy0));
240     P[i + 2] = 200 * (f_xyz2lab(yy0) - f_xyz2lab(zz0));
241 }
242 }
243
244 inline float Conversion::f_xyz2lab(float t) {
245     if (t > 0.008856) {
246         return pow(t, 0.3333333333f);
247     }
248     return 7.787 * t + 0.137931034482759f;
249 }
250
251 /*! Conversion from CIE La*b* to Redness Index
252 \param O a uchar pointer to the source image
253 \param P a uchar pointer to the destination image
254 \param nData an int indicating the total number of pixels
255 */
256 void Conversion::Lab2RI(float *O, float *P, int nData) {
257     uint32_t j = 0;
258     float L, a, b;
259     for (int i = 0; i < nData; i += 3) {
260         L = *(O + i);
261         a = *(O + i + 1);
262         b = *(O + i + 2);
263         P[j++] =
264             (L * (pow((pow(a, 2.0f) + pow(b, 2.0f)), 0.5f) * (
265                 pow(10, 8.2f)))) /
266             (b * pow(L, 6.0f));
267     }
268 }
```

H.0.18 Enhance Class

```

45  void AdaptiveContrastStretch(const Mat &src, Mat &dst,
46      uint8_t kernelsize,
47      float factor);
48
49  void Blur(uint8_t kernelsize, bool chain = false);
50  void Blur(const Mat &src, Mat &dst, uint8_t kernelsize);
51
52  void HistogramEqualization(bool chain = false);
53  void HistogramEqualization(const Mat &src, Mat &dst);
54 }

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8  */
9
10 /*! \class Enhance
11 class which enhances a greyscale cv::Mat image
12 */
13 #include "Enhance.h"
14
15 namespace Vision {
16 /*! Constructor
17 Enhance::Enhance() {}
18
19 /*! Constructor
20 \param src cv::Mat source image
21 */
22 Enhance::Enhance(const Mat &src) {
23     OriginalImg = src;
24     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
25 }
26
27 Enhance::Enhance(const Enhance &rhs) {
28     this->OriginalImg = rhs.OriginalImg;
29     this->ProcessedImg = rhs.OriginalImg;
30     this->TempImg = rhs.TempImg;
31 }
32
33 /*! Constructor
34 \param src cv::Mat source image
35 \param dst cv::Mat destination image
36 \param kernelsize an uchar which represent the kernelsize
37     should be an uneven
38     number higher than two
39 \param factor float which indicates the amount the effect
40     should take place
41 standard value is 1.0 only used in the adaptive contrast
42     stretch enhancement
43 \param operation enumerator EnhanceOperation which
44     enhancement should be

```

```

39 performed
40 */
41 Enhance::Enhance(const Mat &src, Mat &dst, uchar kernelsize,
42     float factor,
43     EnhanceOperation operation) {
44     OriginalImg = src;
45     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
46     switch (operation) {
47     case Vision::Enhance::_AdaptiveContrastStretch:
48         AdaptiveContrastStretch(kernelsize, factor);
49         break;
50     case Vision::Enhance::_Blur:
51         Blur(kernelsize);
52         break;
53     case Vision::Enhance::_HistogramEqualization:
54         HistogramEqualization();
55         break;
56     }
57     dst = ProcessedImg;
58 }
59 /*! Dec-constructor*/
60 Enhance::~Enhance() {}

61 Enhance &Enhance::operator=(Enhance rhs) {
62     if (&rhs != this) {
63         this->OriginalImg = rhs.OriginalImg;
64         this->ProcessedImg = rhs.ProcessedImg;
65         this->TempImg = rhs.ProcessedImg;
66     }
67     return *this;
68 }
69 }

70 /*! Calculate the standard deviation of the neighboring
71     pixels
72 \param 0 uchar pointer to the current pixel of the original
73     image
74 \param i current counter
75 \param hKsize half the kernelsize
76 \param nCols total number of columns
77 \param noNeighboursPix total number of neighboring pixels
78 \param mean mean value of the neighboring pixels
79 \return standard deviation
80 */
81 float Enhance::CalculateStdOfNeighboringPixels(uchar *0, int
82     i, int hKsize,
83                                         int nCols,
84                                         int
85                                         noNeighboursPix
86                                         ,
87                                         float mean) {
88     uint32_t sum_dev = 0.0;
89     float Std = 0.0;
90     sum_dev = 0.0;
91     Std = 0.0;
92     for (int j = -hKsize; j < hKsize; j++) {

```

```

88     for (int k = -hKsize; k < hKsize; k++) {
89         // sum_dev += pow((0[i + j * nCols + k] - mean), 2);
90         sum_dev += SoilMath::quick_pow2((0[i + j * nCols + k]
91                                         - mean));
92     }
93     // Std = sqrt(sum_dev / noNeighboursPix);
94     Std = SoilMath::fastPow(static_cast<double>(sum_dev) /
95                             noNeighboursPix), 2);
96     return Std;
97 }
98 /*! Calculate the sum of the neighboring pixels
99 \param 0 uchar pointer to the current pixel of the original
100    image
101 \param i current counter
102 \param hKsize half the kernelsize
103 \param nCols total number of columns
104 \param sum Total sum of the neighboringpixels
105 */
106 void Enhance::CalculateSumOfNeighboringPixels(uchar *0, int
107                                                 i, int hKsize,
108                                                 int nCols,
109                                                 uint32_t &
110                                                 sum) {
111     for (int j = -hKsize; j < hKsize; j++) {
112         for (int k = -hKsize; k < hKsize; k++) {
113             sum += 0[i + j * nCols + k];
114         }
115     }
116     /*! Homebrew AdaptiveContrastStretch function which
117        calculate the mean and
118        standard deviation from the neighboring pixels if the
119        current pixel is higher
120        then the mean the value is incremented with an given factor
121        multiplied with the
122        standard deviation, and decreased if it's lower then the
123        mean.
124 \param src cv::Mat source image
125 \param dst cv::Mat destination image
126 \param kernelsize an uchar which represent the kernelsize
127        should be an uneven
128        number higher than two
129 \param factor float which indicates the amount the effect
130        should take place
131 \param standard value is 1.0 only used in the adaptive contrast
132        stretch enhancement
133 */
134 void Enhance::AdaptiveContrastStretch(const Mat &src, Mat &
135                                         dst,
136                                         uchar kernelsize,
137                                         float factor) {
138     OriginalImg = src;
139     ProcessedImg.create(OriginalImg.size(), CV_8UC1);

```

```

129     AdaptiveContrastStretch(kernelsize, factor);
130     dst = ProcessedImg;
131 }
132
133 /*! Homebrew AdaptiveContrastStretch function which
134 calculate the mean and
135 standard deviation from the neighboring pixels if the
136 current pixel is higher
137 then the mean the value is incremented with an given factor
138 multiplied with the
139 standard deviation, and decreased if it's lower then the
140 mean.
141 \param kernelsize an uchar which represent the kernelsize
142 should be an uneven
143 number higher than two
144 \param factor float which indicates the amount the effect
145 should take place
146 standard value is 1.0 only used in the adaptive contrast
147 stretch enhancement
148 \param chain use the results from the previous operation
149 default value = false;
150 */
151 void Enhance::AdaptiveContrastStretch(uchar kernelsize,
152                                     float factor,
153                                     bool chain) {
154
155     // Exception handling
156     EMPTY_CHECK(OriginalImg);
157     if (kernelsize < 3 || (kernelsize % 2) == 0) {
158         throw Exception::WrongKernelSizeException();
159     }
160     CV_Assert(OriginalImg.depth() != sizeof(uchar));
161
162     // Make the pointers to the Data
163     uchar *O;
164     CHAIN_PROCESS(chain, 0, uchar);
165     uchar *P = ProcessedImg.data;
166
167     int i = 0;
168     int hKsize = kernelsize / 2;
169     int nCols = OriginalImg.cols;
170     int pStart = (hKsize * nCols) + hKsize + 1;
171
172     int nData = OriginalImg.rows * OriginalImg.cols;
173     int pEnd = nData - pStart;
174     uint32_t noNeighboursPix = kernelsize * kernelsize;
175     uint32_t sum;
176     float mean = 0.0;
177
178     uchar *nRow = GetNRow(nData, hKsize, nCols, OriginalImg.
179                           rows);
180
181     i = pStart;
182     while (i++ < pEnd) {
183         // Checks if pixel isn't a border pixel and progresses
184         // to the new row
185         if (nRow[i] == 1) {

```

```

174         i += kernelsize;
175     }
176
177     // Fill the neighboring pixel array
178     sum = 0;
179     mean = 0;
180
181     // Calculate the statistics
182     CalculateSumOfNeighboringPixels(0, i, hKsize, nCols, sum
183         );
184     mean = (float)(sum / noNeighboursPix);
185     float Std = CalculateStdOfNeighboringPixels(0, i, hKsize
186         , nCols,
187                                         noNeighboursPix
188                                         , mean);
189
190     // Stretch
191
192     if (O[i] > mean) {
193         // int addValue = O[i] + (int)(round(factor * Std));
194         int addValue = O[i] + static_cast<int>(round(factor *
195             Std));
196         if (addValue < 255) {
197             P[i] = addValue;
198         } else {
199             P[i] = 255;
200         }
201     } else if (O[i] < mean) {
202         // int subValue = O[i] - (int)(round(factor * Std));
203         int subValue = O[i] - static_cast<int>(round(factor *
204             Std));
205         if (subValue > 0) {
206             P[i] = subValue;
207         } else {
208             P[i] = 0;
209         }
210     }
211
212     // Stretch the image with an normal histogram equalization
213     HistogramEqualization(true);
214
215
216     /*! Blurs the image with a NxN kernel
217     \param src cv::Mat source image
218     \param dst cv::Mat destination image
219     \param kernelsize an uchar which represent the kernelsize
220         should be an uneven
221         number higher than two
222     */
223     void Enhance::Blur(const Mat &src, Mat &dst, uchar
224     kernelsize) {

```

```

223     OriginalImg = src;
224     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
225     Blur(kernelsize);
226     dst = ProcessedImg;
227 }
228
229 /*! Blurs the image with a NxN kernel
230 \param kernelsize an uchar which represent the kernelsize
231     should be an uneven
232 number higher than two
233 \param chain use the results from the previous operation
234     default value = false;
235 */
236 void Enhance::Blur(uchar kernelsize, bool chain) {
237     // Exception handling
238     EMPTY_CHECK(OriginalImg);
239     if (kernelsize < 3 || (kernelsize % 2) == 0) {
240         throw Exception::WrongKernelSizeException();
241     }
242     CV_Assert(OriginalImg.depth() != sizeof(uchar));
243
244     // Make the pointers to the Data
245     uchar *O;
246     CHAIN_PROCESS(chain, O, uchar);
247     uchar *P = ProcessedImg.data;
248
249     int nData = OriginalImg.rows * OriginalImg.cols;
250     int hKsize = kernelsize / 2;
251     int nCols = OriginalImg.cols;
252     int pStart = (hKsize * nCols) + hKsize + 1;
253     int pEnd = nData - pStart;
254     int noNeighboursPix = kernelsize * kernelsize;
255     uint32_t sum;
256
257     int i;
258     uchar *nRow = GetNRow(nData, hKsize, nCols, OriginalImg.
259                           rows);
260     i = pStart;
261     while (i++ < pEnd) {
262         // Checks if pixel isn't a border pixel and progresses
263         // to the new row
264         if (nRow[i] == 1) {
265             i += kernelsize;
266         }
267
268         // Calculate the sum of the kernel
269         sum = 0;
270         CalculateSumOfNeighboringPixels(0, i, hKsize, nCols, sum
271                                         );
272         P[i] = (uchar)(round(sum / noNeighboursPix));
273     }
274     delete [] nRow;
275 }

```

```
274 /*! Stretches the image using a histogram
275 \param chain use the results from the previous operation
276     default value = false;
277 */
278 void Enhance::HistogramEqualization(bool chain) {
279     // Exception handling
280     EMPTY_CHECK(OriginalImg);
281     CV_Assert(OriginalImg.depth() != sizeof(uchar));
282
283     // Make the pointers to the Data
284     uchar *O;
285     CHAIN_PROCESS(chain, 0, uchar);
286     uchar *P = ProcessedImg.data;
287
288     // Calculate the statics of the whole image
289     ucharStat_t imgStats(0, OriginalImg.rows, OriginalImg.cols
290                         );
291     float sFact;
292     if (imgStats.min != imgStats.max) {
293         sFact = 255.0f / (imgStats.max - imgStats.min);
294     } else {
295         sFact = 1.0f;
296     }
297
298     uint32_t i = 256;
299     uchar LUT_changeValue[256];
300     while (i-- > 0) {
301         LUT_changeValue[i] = (uchar)((float)(i)*sFact) + 0.5f;
302     }
303
304     O = OriginalImg.data;
305     i = OriginalImg.cols * OriginalImg.rows + 1;
306     while (i-- > 0) {
307         *P++ = LUT_changeValue[*O++ - imgStats.min];
308     }
309 }
```

H.0.19 Morphological filter Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11 #define MORPHOLOGICALFILTER_VERSION 1
12
13 namespace Vision {
14 class MorphologicalFilter : public ImageProcessing {
15 public:
16     enum FilterType { OPEN, CLOSE, ERODE, DILATE, NONE };
17
18     MorphologicalFilter();
19     MorphologicalFilter(FilterType filtertype);
20     MorphologicalFilter(const Mat &src, FilterType filtertype
21                         = FilterType::NONE);
22     MorphologicalFilter(const MorphologicalFilter &rhs);
23
24     ~MorphologicalFilter();
25
26     MorphologicalFilter &operator=(MorphologicalFilter &rhs);
27
28     void Dilation(const Mat &mask, bool chain = false);
29     void Erosion(const Mat &mask, bool chain = false);
30
31     void Close(const Mat &mask, bool chain = false);
32     void Open(const Mat &mask, bool chain = false);
33
34 private:
35     void Filter(const Mat &mask, bool chain, uchar startVal,
36                 uchar newVal,
37                 uchar switchVal);
38 };
39 }
```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #include "MorphologicalFilter.h"
11
12 namespace Vision {
```

```
11 MorphologicalFilter::MorphologicalFilter() {}
12
13 MorphologicalFilter::MorphologicalFilter(FilterType
14     filtertype) {
15     switch (filtertype) {
16     case FilterType::OPEN:
17         Open(OriginalImg);
18         break;
19     case FilterType::CLOSE:
20         Close(OriginalImg);
21         break;
22     case FilterType::ERODE:
23         Erosion(OriginalImg);
24         break;
25     case FilterType::DILATE:
26         Dilation(OriginalImg);
27         break;
28     case FilterType::NONE:
29         break;
30     }
31
32 MorphologicalFilter::MorphologicalFilter(const Mat &src,
33                                         FilterType
34                                         filtertype) {
35     OriginalImg = src;
36     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
37     switch (filtertype) {
38     case FilterType::OPEN:
39         Open(OriginalImg);
40         break;
41     case FilterType::CLOSE:
42         Close(OriginalImg);
43         break;
44     case FilterType::ERODE:
45         Erosion(OriginalImg);
46         break;
47     case FilterType::DILATE:
48         Dilation(OriginalImg);
49         break;
50     case FilterType::NONE:
51         break;
52     }
53
54 MorphologicalFilter::MorphologicalFilter(const
55     MorphologicalFilter &rhs) {
56     this->OriginalImg = rhs.OriginalImg;
57     this->ProcessedImg = rhs.ProcessedImg;
58     this->TempImg = rhs.ProcessedImg;
59 }
60 MorphologicalFilter::~MorphologicalFilter() {}
61
62 MorphologicalFilter &MorphologicalFilter::operator=(  
    MorphologicalFilter &rhs) {
```

```

63     if (&rhs != this) {
64         this->OriginalImg = rhs.OriginalImg;
65         this->ProcessedImg = rhs.ProcessedImg;
66         this->TempImg = rhs.TempImg;
67     }
68     return *this;
69 }
70
71 void MorphologicalFilter::Open(const Mat &mask, bool chain)
72 {
73     Erosion(mask, chain);
74     Dilation(mask, true);
75 }
76 void MorphologicalFilter::Close(const Mat &mask, bool chain)
77 {
78     Dilation(mask, chain);
79     Erosion(mask, true);
80 }
81 void MorphologicalFilter::Dilation(const Mat &mask, bool
82     chain) {
83     Filter(mask, chain, 0, 1, 1);
84 }
85 void MorphologicalFilter::Erosion(const Mat &mask, bool
86     chain) {
87     Filter(mask, chain, 1, 0, 0);
88 }
89 void MorphologicalFilter::Filter(const Mat &mask, bool chain
90     , uchar startVal,
91                     uchar newVal, uchar
92                     switchVal) {
93     // Exception handling
94     CV_Assert(OriginalImg.depth() != sizeof(uchar));
95     EMPTY_CHECK(OriginalImg);
96     if (mask.cols % 2 == 0 || mask.cols < 3) {
97         throw Exception::WrongKernelSizeException("Wrong
98             Kernelsize columns!");
99     }
100    if (mask.rows % 2 == 0 || mask.rows < 3) {
101        throw Exception::WrongKernelSizeException("Wrong
102             Kernelsize rows!");
103    }
104    // make Pointers
105    Mat workOrigImg(ProcessedImg.rows + mask.rows,
106                     ProcessedImg.cols + mask.cols,
107                     CV_8UC1);
108    workOrigImg.setTo(0);
109    if (chain) {
110        ProcessedImg.copyTo(workOrigImg(

```

```

110         cv::Rect(hKsizeCol, hKsizeRow, ProcessedImg.cols,
111                     ProcessedImg.rows));
112     // workOrigImg(cv::Rect(hKsizeCol, hKsizeRow,
113     // ProcessedImg.cols,
114     // ProcessedImg.rows)) = ProcessedImg.clone();
115 } else {
116     OriginalImg.copyTo(workOrigImg(
117         cv::Rect(hKsizeCol, hKsizeRow, ProcessedImg.cols,
118                     ProcessedImg.rows)));
119     // workOrigImg(cv::Rect(hKsizeCol, hKsizeRow,
120     // ProcessedImg.cols,
121     // ProcessedImg.rows)) = OriginalImg.clone();
122 }
123 uchar *O = workOrigImg.data;
124
125 // Init the relevant data
126 //uint32_t nData = OriginalImg.cols * OriginalImg.rows;
127 uint32_t nWData = workProcImg.cols * workProcImg.rows;
128 uint32_t nWStart = (hKsizeRow * workProcImg.cols) +
129     hKsizeRow;
130 uint32_t nWEnd = nWData - hKsizeCol - hKsizeRow *
131     workProcImg.cols - 1;
132 uchar *nRow = GetNRow(nWData, hKsizeCol, workProcImg.cols,
133     workProcImg.rows);
134 int MaskPixel = 0, OPixel = 0;
135
136 workProcImg.setTo(0);
137 if (startVal != 0) {
138     workProcImg(cv::Rect(hKsizeCol, hKsizeRow, ProcessedImg.
139     cols,
140                     ProcessedImg.rows)).setTo(startVal)
141     ;
142 }
143 SHOW_DEBUG_IMG(workOrigImg, uchar, 255, "workOrigImg
144     Filter!", false);
145 SHOW_DEBUG_IMG(mask, uchar, 255, "Filter mask", true);
146 for (uint32_t i = nWStart; i < nWEnd; i++) {
147     // Checks if pixel isn't a border pixel and progresses
148     // to the new row
149     if (nRow[i] == 1) {
150         i += mask.cols;
151     }
152     for (int r = 0; r < mask.rows; r++) {
153         for (int c = 0; c < mask.cols; c++) {
154             MaskPixel = c + r * mask.cols;
155             OPixel = i - hKsizeCol + c + (r - hKsizeRow) *
156                 workProcImg.cols;
157             if (mask.data[MaskPixel] == 1 && O[OPixel] ==
158                 switchVal) {
159                 P[i] = newVal;

```

```
152         c = mask.cols;
153         r = mask.rows;
154     }
155 }
156 }
157 }
158 delete [] nRow;
159 SHOW_DEBUG_IMG(workProcImg, uchar, 255, "workProcImg
160     Filter!", true);
160 ProcessedImg = workProcImg(Rect(hKsizeCol, hKsizeRow,
161         ProcessedImg.cols,
162             ProcessedImg.rows)).clone
163     ());
162 SHOW_DEBUG_IMG(ProcessedImg, uchar, 255, "Processed Image
163     Filter!", true);
164 }
```

H.0.20 Segment Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include <vector>
13 #include <queue>
14 #include <string>
15 #include <stdint.h>
16 #include <iostream>
17 #include <algorithm>
18 #include <utility>
19
20 #include <boost/range/adaptor/reversed.hpp>
21
22 #include "ImageProcessing.h"
23 #include "MorphologicalFilter.h"
24 #include "../SoilMath/SoilMath.h"
25
26 namespace Vision {
27 class Segment : public ImageProcessing {
28 public:
29     /*! Coordinates for the region of interest*/
30     typedef struct Rect {
31         uint16_t leftX; /*!< Left X coordinate*/
32         uint16_t leftY; /*!< Left Y coordinate*/
33         uint16_t rightX; /*!< Right X coordinate*/
34         uint16_t rightY; /*!< Right Y coordinate*/
35         Rect(uint16_t lx, uint16_t ly, uint16_t rx, uint16_t ry)
36             : leftX(lx), leftY(ly), rightX(rx), rightY(ry){}
37     } Rect_t;
38
39     typedef std::vector<Vision::Segment::Rect_t> RectList_t;
40
41     /*! Individual blob*/
42     typedef struct Blob {
43         uint16_t Label; /*!< ID of the blob*/
44         cv::Mat Img; /*!< BW image of the blob all the pixel
45             belonging to the blob
46             are set to 1 others are 0*/
47         cv::Rect ROI; /*!< Coordinates for the blob in the
48             original picture as a
49             cv::Rect*/
50         uint32_t Area; /*!< Calculated stats of the blob*/
51         cv::Point<double> Centroid;
52         double Theta;

```

```

51     Blob(uint16_t label, uint32_t area) : Label(label), Area
52         (area){}
53     } Blob_t;
54
55     typedef std::vector<Blob_t> BlobList_t;
56     BlobList_t BlobList; /*!< vector with all the individual
57         blobs*/
58
59     /*! Enumerator to indicate what kind of object to extract
60         */
61     enum TypeOfObjects {
62         Bright, /*!< Enum value Bright object */
63         Dark    /*!< Enum value Dark object. */
64     };
65
66     /*! Enumerator to indicate how the pixel correlate between
67         each other in a
68         * blob*/
69     enum Connected {
70         Four =
71             2, /*!< Enum Four connected, relation between Center
72                 , North, East, South
73                 and West*/
74         Eight =
75             4 /*!< Enum Eight connected, relation between Center
76                 , North, NorthEast,
77                 East, SouthEast, South, SouthWest, West and
78                 NorthWest */
79     };
80
81     /*!< Enumerator which indicate which Segmentation
82         technique should be used */
83     enum SegmentationType {
84         Normal, /*!< Segmentation looking at the intensity of an
85                 individual pixel */
86         LabNeuralNet, /*!< Segmentation looking at the chromatic
87                         a* and b* of the
88                         processed pixel and it's surrounding
89                         pixels, feeding it in
90                         an Neural Net */
91         GraphMinCut /*!< Segmentation using a graph function and
92                         the minimum cut */
93     };
94
95     cv::Mat LabelledImg; /*!< Image with each individual
96         blob labeled with a
97             individual number */
98     uint16_t MaxLabel = 0; /*!< Maximum labels found in the
99         labelled image*/
100    uint16_t noOfFilteredBlobs =
101        0; /*!< Total numbers of blobs that where filtered
102            beacuse the where
103            smaller than the minBlobArea*/
104
105    ucharStat_t OriginalImgStats; /*!< Statistical data from
106        the original image*/

```

```

91     uint8_t ThresholdLevel = 0;      /*!< Current calculated
92         threshold level*/
93
94     float sigma = 2;
95     uint32_t thresholdOffset = 4;
96
97     Segment();
98     Segment(const Mat &src);
99     Segment(const Segment &rhs);
100
101    ~Segment();
102
103    Segment &operator=(Segment &rhs);
104
105    void LoadOriginalImg(const Mat &src);
106
107    void ConvertToBW(TypeOfObjects Typeobjects);
108    void ConvertToBW(const Mat &src, Mat &dst, TypeOfObjects
109        Typeobjects);
110
111    void GetEdges(bool chain = false, Connected conn = Eight);
112    void GetEdges(const Mat &src, Mat &dst, bool chain = false
113        ,
114            Connected conn = Eight);
115
116    void GetEdgesEroding(bool chain = false);
117
118    void GetBlobList(bool chain = false, Connected conn =
119        Eight);
120
121    void Threshold(uchar t, TypeOfObjects Typeobjects);
122
123    void LabelBlobs(bool chain = false, uint16_t minBlobArea =
124        25,
125            Connected conn = Eight);
126
127    private:
128        uint8_t GetThresholdLevel(TypeOfObjects TypeObject);
129        void SetBorder(uchar *P, uchar setValue);
130        void FloodFill(uchar *O, uchar *P, uint16_t x, uint16_t y,
131            uchar fillValue,
132                uchar OldValue);
133        void MakeConsecutive(uint16_t *valueArr, uint32_t noElem,
134            uint16_t &maxlabel);
135        void MakeConsecutive(uint16_t *valueArr, uint16_t *keyArr,
136            uint16_t noElem,
137                uint16_t &maxlabel);
138        void SortAdjacencyList(std::vector<std::vector<uint16_t>>
139            &adj);
140
141        void ConnectedBlobs(uchar *O, uint16_t *P,

```

```
136             std::vector<std::vector<uint16_t>> &
137             adj, uint32_t nCols,
138             uint32_t nRows, Connected conn);
139     void InvertAdjacencyList(std::vector<std::vector<uint16_t
140             >> &adj,
141             std::vector<std::vector<uint16_t
142             >> &adjInv);
143 }
144 }
```

```
1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (Jelle Spijker)
5  * This software is proprietary and confidential
6  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
7  */
8 /*! \class Segment
9 \brief Segmentation algorithms
10 With this class, various segmentation routines can be
11 applied to a greyscale or
12 black and white source image.
13 */
14 #include "Segment.h"
15
16 namespace Vision {
17 //! Constructor of the Segmentation class
18 Segment::Segment() {}
19
20 //! Constructor of the Segmentation class
21 Segment::Segment(const Mat &src) {
22     OriginalImg = src;
23     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
24     LabelledImg.create(OriginalImg.size(), CV_16UC1);
25 }
26
27 Segment::Segment(const Segment &rhs) {
28     this->BlobList = rhs.BlobList;
29     this->LabelledImg = rhs.LabelledImg;
30     this->MaxLabel = rhs.MaxLabel;
31     this->noOfFilteredBlobs = rhs.noOfFilteredBlobs;
32     this->OriginalImg = rhs.OriginalImg;
33     this->OriginalImgStats = rhs.OriginalImgStats;
34     this->ProcessedImg = rhs.ProcessedImg;
35     this->TempImg = rhs.TempImg;
36     this->ThresholdLevel = rhs.ThresholdLevel;
37 }
38
39 //! De-constructor
40 Segment::~Segment() {}
41 Segment &Segment::operator=(Segment &rhs) {
42     if (&rhs != this) {
43         this->BlobList = rhs.BlobList;
```

```

44     this->LabelledImg = rhs.LabelledImg;
45     this->MaxLabel = rhs.MaxLabel;
46     this->noOfFilteredBlobs = rhs.noOfFilteredBlobs;
47     this->OriginalImg = rhs.OriginalImg;
48     this->OriginalImgStats = rhs.OriginalImgStats;
49     this->ProcessedImg = rhs.ProcessedImg;
50     this->TempImg = rhs.TempImg;
51     this->ThresholdLevel = rhs.ThresholdLevel;
52 }
53 return *this;
54 }
55
56 void Segment::LoadOriginalImg(const Mat &src) {
57     OriginalImg = src;
58     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
59     LabelledImg.create(OriginalImg.size(), CV_16UC1);
60 }
61
62 /*! Determine the threshold level by iteration, between two
   distribution,
63 presumably back- and foreground. It works towards the
   average of the two
64 averages and finally sets the threshold with two time the
   standard deviation
65 from the mean of the set object
66 \param TypeOfObject is an enumerator indicating if the bright
   or the dark pixels
67 are the object and should be set to one
68 \return The threshold level as an uint8_t */
69 uint8_t Segment::GetThresholdLevel(TypeOfObjects TypeOfObject)
70 {
71     // Exception handling
72     EMPTY_CHECK(OriginalImg);
73     CV_Assert(OriginalImg.depth() != sizeof(uchar));
74
75     // Calculate the statistics of the whole picture
76     ucharStat_t OriginalImgStats(OriginalImg.data, OriginalImg
77                                 .rows,
78                                 OriginalImg.cols);
79
80     // Sets the initial threshold with the mean of the total
81     // picture
82     pair<uchar, uchar> T;
83     T.first = (uchar)(OriginalImgStats.Mean + 0.5);
84     T.second = 0;
85
86     uchar Rstd = 0;
87     uchar Lstd = 0;
88     uchar Rmean = 0;
89     uchar Lmean = 0;
90
91     // Iterate till optimum Threshold is found between back- &
92     // foreground
93     while (T.first != T.second) {
94         // Gets an array of the left part of the histogram
95         uint32_t i = T.first;

```

```

92     uint32_t *Left = new uint32_t[i]{};
93     while (i-- > 0) {
94         Left[i] = OriginalImgStats.bins[i];
95     }
96
97     // Gets an array of the right part of the histogram
98     uint32_t rightEnd = 256 - T.first;
99     uint32_t *Right = new uint32_t[rightEnd]{};
100    i = rightEnd;
101    while (i-- > 0) {
102        Right[i] = OriginalImgStats.bins[i + T.first];
103    }
104
105    // Calculate the statistics of both histograms,
106    // taking into account the current threshold
107    ucharStat_t sLeft(Left, 0, T.first);
108    ucharStat_t sRight(Right, T.first, 256);
109
110    // Calculate the new threshold the mean of the means
111    T.second = T.first;
112    T.first = (uchar)((sLeft.Mean + sRight.Mean) / 2) +
113        0.5;
114
115    Rmean = (uchar)(sRight.Mean + 0.5);
116    Lmean = (uchar)(sLeft.Mean + 0.5);
117    Rstd = (uchar)(sRight.Std + 0.5);
118    Lstd = (uchar)(sLeft.Std + 0.5);
119    delete[] Left;
120    delete[] Right;
121 }
122
123 // Assumes the pixel value of the sought object lies
124 // between 2 sigma
125 int val = 0;
126 switch (TypeObject) {
127 case Bright:
128     val = Rmean - (sigma * Rstd) - thresholdOffset;
129     if (val < 0) {
130         val = 0;
131     } else if (val > 255) {
132         val = 255;
133     }
134     T.first = (uchar)val;
135     break;
136 case Dark:
137     val = Lmean + (sigma * Lstd) + thresholdOffset;
138     if (val < 0) {
139         val = 0;
140     } else if (val > 255) {
141         val = 255;
142     }
143     T.first = (uchar)val;
144     break;
145 }
146
147 return T.first;

```

```

146 }
147
148 /*! Convert a greyscale image to a BW using an automatic
149   Threshold
150 \param src is the source image as a cv::Mat
151 \param dst destination image as a cv::Mat
152 \param TypeObject is an enumerator indicating if the bright
153   or the dark pixels
154 are the object and should be set to one */
155 void Segment::ConvertToBW(const Mat &src, Mat &dst,
156   TypeOfObjects Typeobjects) {
157   OriginalImg = src;
158   ProcessedImg.create(OriginalImg.size(), CV_8UC1);
159   LabelledImg.create(OriginalImg.size(), CV_16UC1);
160   ConvertToBW(Typeobjects);
161   dst = ProcessedImg;
162 }
163
164 /*! Convert a greyscale image to a BW using an automatic
165   Threshold
166 \param TypeObject is an enumerator indicating if the bright
167   or the dark pixels
168 are the object and should be set to one */
169 void Segment::ConvertToBW(TypeOfObjects Typeobjects) {
170   // Determine the threshold
171   uchar T = GetThresholdLevel(Typeobjects);
172   // Threshold the picture
173   Threshold(T, Typeobjects);
174 }
175
176 /*! Convert a greyscale image to a BW
177 \param t uchar set the value which is the tipping point
178 \param TypeObject is an enumerator indicating if the bright
179   or the dark pixels
180 are the object and should be set to one */
181 void Segment::Threshold(uchar t, TypeOfObjects Typeobjects)
182 {
183   // Exception handling
184   EMPTY_CHECK(OriginalImg);
185   CV_Assert(OriginalImg.depth() != sizeof(uchar) ||
186             OriginalImg.depth() != sizeof(uint16_t));
187
188   // Create LUT
189   uchar LUT(newValue[256]{0};
190   if (Typeobjects == Bright) {
191     for (uint32_t i = t; i < 256; i++) {
192       LUT(newValue[i] = 1;
193     }
194   } else {
195     for (uint32_t i = 0; i <= t; i++) {
196       LUT(newValue[i] = 1;
197     }
198   }
199
200   // Create the pointers to the data

```

```
195     uchar *P = ProcessedImg.data;
196     uchar *O = OriginalImg.data;
197
198     // Fills the ProcessedImg with either a 0 or 1
199     for (int i = 0; i < OriginalImg.cols * OriginalImg.rows; i
200        ++) {
201         P[i] = LUT_newValue[O[i]];
202     }
203
204     /*! Set all the border pixels to a set value
205     \param *P uchar pointer to the Mat.data
206     \param setValue uchar the value which is written to the
207         border pixels
208 */
209     void Segment::SetBorder(uchar *P, uchar setValue) {
210         // Exception handling
211         EMPTY_CHECK(OriginalImg);
212         CV_Assert(OriginalImg.depth() != sizeof(uchar) ||
213                 OriginalImg.depth() != sizeof(uint16_t));
214
215         uint32_t nData = OriginalImg.cols * OriginalImg.rows;
216
217         // Set borderPixels to 2
218         uint32_t i = 0;
219         uint32_t pEnd = OriginalImg.cols + 1;
220
221         // Set the top row to value 2
222         while (i < pEnd) {
223             P[i++] = setValue;
224         }
225
226         // Set the bottom row to value 2
227         i = nData + 1;
228         pEnd = nData - OriginalImg.cols;
229         while (i-- > pEnd) {
230             P[i] = setValue;
231         }
232
233         // Sets the first and the last Column to 2
234         i = 1;
235         pEnd = OriginalImg.rows;
236         while (i < pEnd) {
237             P[(i * OriginalImg.cols) - 1] = setValue;
238             P[(i++ * OriginalImg.cols)] = setValue;
239         }
240
241     /*! Remove the blobs that are connected to the border
242     \param conn set the pixel connection eight or four
243     \param chain use the results from the previous operation
244         default value = false;
245 */
246     void Segment::RemoveBorderBlobs(uint32_t border, bool chain)
247     {
248         CV_Assert(OriginalImg.depth() != sizeof(uchar));
```

```

247     EMPTY_CHECK(OriginalImg);
248     // make Pointers
249     uchar *O;
250     CHAIN_PROCESS(chain, 0, uchar);
251     if (chain) {
252         ProcessedImg = TempImg.clone();
253     } else {
254         ProcessedImg = OriginalImg.clone();
255     }
256
257     SHOW_DEBUG_IMG(OriginalImg, uchar, 255, "Original Image
258                     RemoverBorderBlobs!",
259                     true);
260     SHOW_DEBUG_IMG(TempImg, uchar, 255, "Temp Image
261                     RemoverBorderBlobs!", true);
262
263     uchar *P = ProcessedImg.data;
264     uint32_t cols = ProcessedImg.cols;
265     uint32_t rows = ProcessedImg.rows;
266
267     try {
268         for (uint32_t i = 0; i < border; i++) {
269             for (uint32_t j = 0; j < cols; j++) {
270                 if (O[(i * cols) + j] == 1 && P[(i * cols) + j] !=
271                     2) {
272                     cv::floodFill(ProcessedImg, cv::Point(j, i), (
273                         uchar)2);
274                 }
275             }
276         }
277         for (uint32_t i = rows - border - 1; i < rows; i++) {
278             for (uint32_t j = 0; j < cols; j++) {
279                 if (O[(i * cols) + j] == 1 && P[(i * cols) + j] !=
280                     2) {
281                     cv::floodFill(ProcessedImg, cv::Point(j, i), (
282                         uchar)2);
283                 }
284             }
285         }
286         for (uint32_t i = border; i < rows - border; i++) {
287             for (uint32_t j = 0; j < border; j++) {
288                 if (O[(i * cols) + j] == 1 && P[(i * cols) + j] !=
289                     2) {
290                     cv::floodFill(ProcessedImg, cv::Point(j, i), (
291                         uchar)2);
292                 }
293             }
294         }
295     } catch (cv::Exception &e) {

```

```

294     }
295     SHOW_DEBUG_IMG(ProcessedImg, uchar, 255,
296                     "Processed Image RemoverBorderBlobs before
297                     LUT!", true);
298
299     // Change values 2 -> 0
300     uchar LUT(newValue[3]{0, 1, 0};
301     P = ProcessedImg.data;
302     uint32_t nData = rows * cols;
303     for (uint32_t i = 0; i < nData; i++) {
304         P[i] = LUT(newValue[P[i]]);
305     }
306
307     SHOW_DEBUG_IMG(ProcessedImg, uchar, 255,
308                     "Processed Image RemoverBorderBlobs!", true
309                     );
310
311     /*! Label all the individual blobs in a BW source image. The
312        result are written
313        to the labelledImg as an ushort
314        \param conn set the pixel connection eight or four
315        \param chain use the results from the previous operation
316        default value = false;
317        \param minBlobArea minimum area when an artifact is
318        considered a blob
319
320     */
321     void Segment::LabelBlobs(bool chain, uint16_t minBlobArea,
322                             Connected conn) {
323         // Exception handling
324         CV_Assert(OriginalImg.depth() != sizeof(uchar));
325         EMPTY_CHECK(OriginalImg);
326
327         // make the Pointers to the data
328         uchar *O;
329         if (chain) {
330             TempImg = ProcessedImg.clone();
331             ProcessedImg = cv::Mat(OriginalImg.rows, OriginalImg.
332                                   cols, CV_16UC1);
333             O = (uchar *)TempImg.data;
334         } else {
335             O = (uchar *)OriginalImg.data;
336         }
337         uint16_t *P = (uint16_t *)LabelledImg.data;
338
339         uint32_t nCols = OriginalImg.cols;
340         uint32_t nRows = OriginalImg.rows;
341         uint32_t nData = nCols * nRows;
342
343         vector<vector<uint16_t>> CLdownstream;
344
345         ConnectedBlobs(O, P, CLdownstream, nCols, nRows,
346                         conn); // First loop through the image
347         SortAdjacencyList(
348             CLdownstream); // Sort all the adjacencylists and make
349                         unique,

```

```

342
343 // identify all the lowest values in the adjacent list
344 uint16_t *valueArr = new uint16_t[CLdownstream.size()];
345 for (int i = CLdownstream.size() - 1; i >= 0; --i) {
346     std::vector<uint16_t *> route;
347     uint16_t minVal = i;
348
349     for (uint32_t j = 0; j < CLdownstream[i].size(); j++) {
350
351         // add the first node to the queue;
352         route.push_back(&CLdownstream[i][j]);
353
354         // iterate till the last node
355         bool lastNodeReached = false;
356         while (!lastNodeReached) {
357             uint32_t nodesVisited = route.size() - 1;
358             if (*route[nodesVisited] < minVal) {
359                 minVal = *route[nodesVisited];
360             }
361             route.push_back(&CLdownstream[*route[nodesVisited
362 ]][0]);
363             if (route[nodesVisited] == route[nodesVisited + 1])
364             {
365                 route.pop_back();
366                 lastNodeReached = true;
367             }
368             // Set all values to the lowest value
369             for (uint32_t k = 0; k < route.size(); k++) {
370                 *route[k] = minVal;
371             }
372             valueArr[i] = minVal;
373         }
374
375         // Make numbers consecutive
376         MakeConsecutive(valueArr, CLdownstream.size(), MaxLabel);
377
378         // Second loop through the pixels to give the values a
379         // final value
380         for_each(P, P + nData, [&](uint16_t &V) { V = valueArr[V];
381             });
382         delete[] valueArr;
383     }
384
385     /*! Create a BW image with only edges from a BW image
386     \param src source image as a const cv::Mat
387     \param dst destination image as a cv::Mat
388     \param conn set the pixel connection eight or four
389     \param chain use the results from the previous operation
390     default value = false;
391 */
392     void Segment::GetEdges(const Mat &src, Mat &dst, bool chain,
393     Connected conn) {
394     OriginalImg = src;
395     GetEdges(chain, conn);

```

```
392     dst = ProcessedImg;
393 }
394
395 /*! Create a BW image with only edges from a BW image
396 \param conn set the pixel connection eight or four
397 \param chain use the results from the previous operation
398     default value = false;
399 */
400 void Segment::GetEdges(bool chain, Connected conn) {
401     // Exception handling
402     CV_Assert(OriginalImg.depth() != sizeof(uchar));
403     EMPTY_CHECK(OriginalImg);
404
405     // make Pointers
406     uchar *O;
407     CHAIN_PROCESS(chain, 0, uchar);
408     uchar *P = ProcessedImg.data;
409
410     uint32_t nCols = OriginalImg.cols;
411     uint32_t nRows = OriginalImg.rows;
412     uint32_t nData = nCols * nRows;
413     uint32_t pEnd = nData + 1;
414     uint32_t i = 0;
415
416     // Loop through the image and set each pixel which has a
417     // zero neighbor set it
418     // to two.
419     if (conn == Four) {
420         // Loop through the picture
421         while (i < pEnd) {
422             // If current value = zero processed value = zero
423             if (O[i] == 0) {
424                 P[i] = 0;
425             }
426             // If current value = 1 check North West, South and
427             // East and act
428             // accordingly
429             else if (O[i] == 1) {
430                 uchar *nPixels = new uchar[4];
431                 nPixels[0] = O[i - 1];
432                 nPixels[1] = O[i - nCols];
433                 nPixels[2] = O[i + 1];
434                 nPixels[3] = O[i + nCols];
435
436                 // Sort the neighbors for easier checking
437                 SoilMath::Sort::QuickSort<uchar>(nPixels, 4);
438                 if (nPixels[0] == 0) {
439                     P[i] = 1;
440                 } else {
441                     P[i] = 0;
442                 }
443                 delete[] nPixels;
444             } else {
445                 throw Exception::PixelValueOutOfBoundsException();
446             }
447             i++;
448         }
449     }
450 }
```

```

445     }
446 } else {
447     // Loop through the picture
448     while (i < pEnd) {
449         // If current value = zero processed value = zero
450         if (O[i] == 0) {
451             P[i] = 0;
452         }
453         // If current value = 1 check North West , South and
454         // East and act
455         // accordingly
456         else if (O[i] == 1) {
457             uchar *nPixels = new uchar[8];
458             nPixels[0] = O[i - 1];
459             nPixels[1] = O[i - nCols];
460             nPixels[2] = O[i - nCols - 1];
461             nPixels[3] = O[i - nCols + 1];
462             nPixels[4] = O[i + 1];
463             nPixels[5] = O[i + nCols + 1];
464             nPixels[6] = O[i + nCols];
465             nPixels[7] = O[i + nCols - 1];
466             // Sort the neighbors for easier checking
467             SoilMath::Sort::QuickSort<uchar>(nPixels, 8);
468
469             if (nPixels[0] == 0) {
470                 P[i] = 1;
471             } else {
472                 P[i] = 0;
473             }
474             delete[] nPixels;
475         } else {
476             throw Exception::PixelValueOutOfBoundsException();
477         }
478         i++;
479     }
480 }
481 }
482
483 void Segment::GetEdgesEroding(bool chain) {
484     // Exception handling
485     CV_Assert(OriginalImg.depth() != sizeof(uchar));
486     EMPTY_CHECK(OriginalImg);
487
488     // make Pointers
489     uchar *O;
490     CHAIN_PROCESS(chain, O, uchar);
491     uchar *P = ProcessedImg.data;
492
493     uint32_t nCols = OriginalImg.cols;
494     uint32_t nRows = OriginalImg.rows;
495     uint32_t nData = nCols * nRows;
496
497     // Setup the erosion
498     MorphologicalFilter eroder;
499     if (chain) {

```

```

500     eroder.OriginalImg = TempImg;
501 } else {
502     eroder.OriginalImg = OriginalImg;
503 }
504 // Setup the processed image of the eroder
505 eroder.ProcessedImg.create(OriginalImg.size(), CV_8UC1);
506 eroder.ProcessedImg.setTo(0);
507 // Setup the mask
508 Mat mask(3, 3, CV_8UC1, 1);
509 // Erode the image
510 eroder.Erosion(mask, false);
511
512 // Loop through the image and set the not eroded pixels to
513 // zero
514 for (uint32_t i = 0; i < nData; i++) {
515     if (O[i] != eroder.ProcessedImg.data[i]) {
516         P[i] = 1;
517     } else {
518         P[i] = 0;
519     }
520 }
521 // ProcessedImg = OriginalImg.clone() - eroder.
522 // ProcessedImg.clone();
523 SHOW_DEBUG_IMG(eroder.ProcessedImg, uchar, 255, "Eroded
524         img Processed Image!",
525         true);
525 SHOW_DEBUG_IMG(ProcessedImg, uchar, 255, "GetEdgesEroding
526         Processed Image!",
527         true);
528 }
529 /*! Create a BlobList subtracting each individual blob out
530 of a Labelled image.
531 If the labelled image is empty build a new one with a BW
532 image.
533 \param conn set the pixel connection eight or four
534 \param chain use the results from the previous operation
535     default value = false;
536 */
537 void Segment::GetBlobList(bool chain, Connected conn) {
538     // Exception handling
539     CV_Assert(OriginalImg.depth() != sizeof(uchar));
540     EMPTY_CHECK(OriginalImg);
541
542     // If there isn't a labelledImg make one
543     if (MaxLabel < 1) {
544         LabelBlobs(chain, 5, conn);
545     }
546
547     // Make an empty BlobList
548     uint32_t nCols = OriginalImg.cols;
549     uint32_t nRows = OriginalImg.rows;
550     uint32_t nData = nCols * nRows;
551     RectList_t rectList;

```

```

549
550 // Calculate Stats the statistics
551 uint16Stat_t LabelStats((uint16_t *)LabelledImg.data,
552                         LabelledImg.cols,
553                         LabelledImg.rows, MaxLabel + 1, 0,
554                         MaxLabel);
555
556 BlobList.reserve(LabelStats.EndBin);
557 rectList.reserve(LabelStats.EndBin);
558
559 BlobList.push_back(Blob_t(0, 0));
560 rectList.push_back(Rect_t(0, 0, 0, 0));
561
562 for (uint32_t i = 1; i < LabelStats.EndBin; i++) {
563     BlobList.push_back(Blob_t(i, LabelStats.bins[i]));
564     rectList.push_back(Rect_t(nCols, nRows, 0, 0));
565 }
566
567 // make Pointers
568 uint16_t *L = (uint16_t *)LabelledImg.data;
569
570 uint32_t currentX, currentY;
571 // uint16_t leftX, leftY, rightX, rightY;
572 // Loop through the labeled image and extract the Blobs
573 for (uint32_t i = 0; i < nData; i++) {
574     if (L[i] != 0) {
575         /* Determine the current x and y value of the current
576          blob and
577          checks if it is min/max */
578         currentY = i / nCols;
579         currentX = i % nCols;
580
581         // Min value
582         if (currentX < rectList[L[i]].leftX) {
583             rectList[L[i]].leftX = currentX;
584         }
585         if (currentY < rectList[L[i]].leftY) {
586             rectList[L[i]].leftY = currentY;
587         }
588
589         // Max value
590         if (currentX > rectList[L[i]].rightX) {
591             rectList[L[i]].rightX = currentX;
592         }
593         if (currentY > rectList[L[i]].rightY) {
594             rectList[L[i]].rightY = currentY;
595         }
596     }
597
598 // Loop through the BlobList and finalize it
599 uint8_t *LUT_filter = new uint8_t[MaxLabel + 1] {};
600 for (uint32_t i = 1; i <= MaxLabel; i++) {
601     LUT_filter[i] = 1;
602     BlobList[i].ROI.y = rectList[i].leftY;
603     BlobList[i].ROI.x = rectList[i].leftX;

```

```

602     BlobList[i].ROI.height = rectList[i].rightY - rectList[i
603         ].leftY + 1;
604     BlobList[i].ROI.width = rectList[i].rightX - rectList[i
605         ].leftX + 1;
606     BlobList[i].Img = CopyMat<uint8_t, uint16_t>(
607         LabelledImg(BlobList[i].ROI).clone(), LUT_filter,
608         CV_8UC1);
609     //SHOW_DEBUG_IMG(BlobList[i].Img, uchar, 255, "Blob",
610         true);
611     LUT_filter[i] = 0;
612 }
613 delete[] LUT_filter;
614
615 // Remove background blob
616 BlobList.erase(BlobList.begin());
617 }
618
619 void Segment::FillHoles(bool chain) {
620     // Exception handling
621     CV_Assert(OriginalImg.depth() != sizeof(uchar));
622     EMPTY_CHECK(OriginalImg);
623
624     // make Pointers
625     uchar *O;
626     CHAIN_PROCESS(chain, O, uchar);
627     if (chain) {
628         ProcessedImg = TempImg.clone();
629     } else {
630         ProcessedImg = OriginalImg.clone();
631     }
632
633     uchar *P = ProcessedImg.data;
634
635     // Determine the starting point of the floodfill
636     int itt = -1;
637     while (P[++itt] != 0)
638         ;
639     uint16_t row = static_cast<uint16_t>(itt / OriginalImg.
640         rows);
641     uint16_t col = static_cast<uint16_t>(itt % OriginalImg.
642         rows);
643
644     // Fill the outside
645     try {
646         cv::floodFill(ProcessedImg, cv::Point(col, row), cv::
647             Scalar(2));
648     } catch (cv::Exception &e) {
649     }
650
651     // Set the unreached areas to 1 and the outside to 0;
652     uchar LUT newVal[3] = {1, 1, 0};
653     uint32_t nData = OriginalImg.rows * OriginalImg.cols;
654     uint32_t i = 0;
655     while (i <= nData) {
656         P[i] = LUT newVal[P[i]];
657         i++;
658     }
659 }
```

```

651     }
652 }
653
654 /*!
655  * \brief Segment::SortAdjacencyList Sort the the sub
656  * vectors
657  * \param adj std::vector<std::vector<uint16_t>> &adj
658  */
659 void Segment::SortAdjacencyList(std::vector<std::vector<
660     uint16_t>> &adj) {
661     uint32_t j = 0;
662     for_each(adj.begin(), adj.end(), [&](std::vector<uint16_t>
663         &L) {
664         std::sort(L.begin(), L.end());
665         std::vector<uint16_t>::iterator it;
666         it = std::unique(L.begin(), L.end());
667         L.resize(std::distance(L.begin(), it));
668         if (L.size() > 1) {
669             for (std::vector<uint16_t>::iterator iter = L.begin();
670                  iter != L.end();
671                  ++iter) {
672                 if (*iter == j) {
673                     L.erase(iter);
674                     break;
675                 }
676             }
677         }
678     }
679     /*!
680  * \brief Segment::ConnectedBlobs Connect all the blobs and
681  * created the
682  * adjacency list
683  * \param O
684  * \param P
685  * \param adj
686  * \param nCols
687  * \param nRows
688  * \param conn
689  */
690 void Segment::ConnectedBlobs(uchar *O, uint16_t *P,
691                             std::vector<std::vector<
692                                 uint16_t>> &adj,
693                             uint32_t nCols, uint32_t nRows,
694                             Connected conn) {
695     // Determine the size of the array for beginning and
696     // endrow and middle of a
697     // row
698     uint32_t noConn[3] = {static_cast<uint32_t>(conn),
699                           (static_cast<uint32_t>(conn) / 2),
700                           (static_cast<uint32_t>(conn) / 2) +
701                           1};
701     uint32_t lastConn[3] = {noConn[0] - 1, noConn[1] - 1,
702                           noConn[2] - 1};

```

```

697     uint32_t nData = nCols * nRows;
698
699     uint16_t currentlbl = 0;
700     vector<uint16_t> zeroVector;
701     zeroVector.push_back(currentlbl);
702     adj.push_back(zeroVector);
703
704     // Determine which borderpixels should be handled
705     // differently
706     uchar *nRow = new uchar[nData]{};
707     for (uint32_t i = nCols; i < nData; i += nCols) {
708         nRow[i] = 1;
709         nRow[i - 1] = 2;
710     }
711
712     // Set the first pixel
713     if (0[0] == 0) {
714         P[0] = 0;
715     } else if (0[0] == 1) {
716         P[0] = 1;
717     } else {
718         throw Exception::PixelValueOutOfBoundsException();
719     }
720
721     // Walk through the toprow and determine if it's a new
722     // blob or it's connected
723     // with previously determine blob
724     for (uint32_t i = 1; i < nCols; i++) {
725         if (0[i] == 0) {
726             P[i] = 0;
727         } else if (0[i] == 1) {
728             // If West is zero assume this is a new blob
729             if (P[i - 1] == 0) {
730                 P[i] = ++currentlbl;
731                 vector<uint16_t> cVector;
732                 cVector.push_back(currentlbl);
733                 adj.push_back(cVector);
734             } else { // set as previous blob
735                 P[i] = P[i - 1];
736             }
737         } else { // Value of of bounds
738             throw Exception::PixelValueOutOfBoundsException();
739         }
740     }
741
742     // walk through each pixel and determine if it's a new
743     // blob or it's connected
744     // with previously determine blob
745     for (uint32_t i = OriginalImg.cols; i < nData; i++) {
746         if (0[i] == 0) { // Original pixel = 0
747             P[i] = 0;
748         } else if (0[i] == 1) {
749             // Get an array of Neighboring Pixels
750             uint16_t *nPixels = new uint16_t[noConn[nRow[i]]];
751             if (nRow[i] != 1) {
752                 nPixels[0] = P[i - 1];
753             }
754             if (i < nData - nCols) {
755                 nPixels[1] = P[i + 1];
756             }
757             if (i > nData - nCols - 1) {
758                 nPixels[2] = P[i - nCols - 1];
759             }
760             if (i > nData - nCols - 2) {
761                 nPixels[3] = P[i - nCols];
762             }
763             if (i < nData - nCols - 2) {
764                 nPixels[4] = P[i + nCols];
765             }
766             if (i < nData - nCols - 1) {
767                 nPixels[5] = P[i + nCols - 1];
768             }
769             if (i > nData - nCols) {
770                 nPixels[6] = P[i - nCols];
771             }
772             if (i > nData - nCols - 3) {
773                 nPixels[7] = P[i - nCols - 2];
774             }
775             if (i > nData - nCols - 4) {
776                 nPixels[8] = P[i - nCols - 3];
777             }
778             if (i < nData - nCols + 1) {
779                 nPixels[9] = P[i + nCols + 1];
780             }
781             if (i < nData - nCols + 2) {
782                 nPixels[10] = P[i + nCols + 2];
783             }
784             if (i < nData - nCols + 3) {
785                 nPixels[11] = P[i + nCols + 3];
786             }
787             if (i < nData - nCols + 4) {
788                 nPixels[12] = P[i + nCols + 4];
789             }
790             if (i < nData - nCols + 5) {
791                 nPixels[13] = P[i + nCols + 5];
792             }
793             if (i < nData - nCols + 6) {
794                 nPixels[14] = P[i + nCols + 6];
795             }
796             if (i < nData - nCols + 7) {
797                 nPixels[15] = P[i + nCols + 7];
798             }
799             if (i < nData - nCols + 8) {
800                 nPixels[16] = P[i + nCols + 8];
801             }
802             if (i < nData - nCols + 9) {
803                 nPixels[17] = P[i + nCols + 9];
804             }
805             if (i < nData - nCols + 10) {
806                 nPixels[18] = P[i + nCols + 10];
807             }
808             if (i < nData - nCols + 11) {
809                 nPixels[19] = P[i + nCols + 11];
810             }
811             if (i < nData - nCols + 12) {
812                 nPixels[20] = P[i + nCols + 12];
813             }
814             if (i < nData - nCols + 13) {
815                 nPixels[21] = P[i + nCols + 13];
816             }
817             if (i < nData - nCols + 14) {
818                 nPixels[22] = P[i + nCols + 14];
819             }
820             if (i < nData - nCols + 15) {
821                 nPixels[23] = P[i + nCols + 15];
822             }
823             if (i < nData - nCols + 16) {
824                 nPixels[24] = P[i + nCols + 16];
825             }
826             if (i < nData - nCols + 17) {
827                 nPixels[25] = P[i + nCols + 17];
828             }
829             if (i < nData - nCols + 18) {
830                 nPixels[26] = P[i + nCols + 18];
831             }
832             if (i < nData - nCols + 19) {
833                 nPixels[27] = P[i + nCols + 19];
834             }
835             if (i < nData - nCols + 20) {
836                 nPixels[28] = P[i + nCols + 20];
837             }
838             if (i < nData - nCols + 21) {
839                 nPixels[29] = P[i + nCols + 21];
840             }
841             if (i < nData - nCols + 22) {
842                 nPixels[30] = P[i + nCols + 22];
843             }
844             if (i < nData - nCols + 23) {
845                 nPixels[31] = P[i + nCols + 23];
846             }
847             if (i < nData - nCols + 24) {
848                 nPixels[32] = P[i + nCols + 24];
849             }
850             if (i < nData - nCols + 25) {
851                 nPixels[33] = P[i + nCols + 25];
852             }
853             if (i < nData - nCols + 26) {
854                 nPixels[34] = P[i + nCols + 26];
855             }
856             if (i < nData - nCols + 27) {
857                 nPixels[35] = P[i + nCols + 27];
858             }
859             if (i < nData - nCols + 28) {
860                 nPixels[36] = P[i + nCols + 28];
861             }
862             if (i < nData - nCols + 29) {
863                 nPixels[37] = P[i + nCols + 29];
864             }
865             if (i < nData - nCols + 30) {
866                 nPixels[38] = P[i + nCols + 30];
867             }
868             if (i < nData - nCols + 31) {
869                 nPixels[39] = P[i + nCols + 31];
870             }
871             if (i < nData - nCols + 32) {
872                 nPixels[40] = P[i + nCols + 32];
873             }
874             if (i < nData - nCols + 33) {
875                 nPixels[41] = P[i + nCols + 33];
876             }
877             if (i < nData - nCols + 34) {
878                 nPixels[42] = P[i + nCols + 34];
879             }
880             if (i < nData - nCols + 35) {
881                 nPixels[43] = P[i + nCols + 35];
882             }
883             if (i < nData - nCols + 36) {
884                 nPixels[44] = P[i + nCols + 36];
885             }
886             if (i < nData - nCols + 37) {
887                 nPixels[45] = P[i + nCols + 37];
888             }
889             if (i < nData - nCols + 38) {
890                 nPixels[46] = P[i + nCols + 38];
891             }
892             if (i < nData - nCols + 39) {
893                 nPixels[47] = P[i + nCols + 39];
894             }
895             if (i < nData - nCols + 40) {
896                 nPixels[48] = P[i + nCols + 40];
897             }
898             if (i < nData - nCols + 41) {
899                 nPixels[49] = P[i + nCols + 41];
900             }
901             if (i < nData - nCols + 42) {
902                 nPixels[50] = P[i + nCols + 42];
903             }
904             if (i < nData - nCols + 43) {
905                 nPixels[51] = P[i + nCols + 43];
906             }
907             if (i < nData - nCols + 44) {
908                 nPixels[52] = P[i + nCols + 44];
909             }
910             if (i < nData - nCols + 45) {
911                 nPixels[53] = P[i + nCols + 45];
912             }
913             if (i < nData - nCols + 46) {
914                 nPixels[54] = P[i + nCols + 46];
915             }
916             if (i < nData - nCols + 47) {
917                 nPixels[55] = P[i + nCols + 47];
918             }
919             if (i < nData - nCols + 48) {
920                 nPixels[56] = P[i + nCols + 48];
921             }
922             if (i < nData - nCols + 49) {
923                 nPixels[57] = P[i + nCols + 49];
924             }
925             if (i < nData - nCols + 50) {
926                 nPixels[58] = P[i + nCols + 50];
927             }
928             if (i < nData - nCols + 51) {
929                 nPixels[59] = P[i + nCols + 51];
930             }
931             if (i < nData - nCols + 52) {
932                 nPixels[60] = P[i + nCols + 52];
933             }
934             if (i < nData - nCols + 53) {
935                 nPixels[61] = P[i + nCols + 53];
936             }
937             if (i < nData - nCols + 54) {
938                 nPixels[62] = P[i + nCols + 54];
939             }
940             if (i < nData - nCols + 55) {
941                 nPixels[63] = P[i + nCols + 55];
942             }
943             if (i < nData - nCols + 56) {
944                 nPixels[64] = P[i + nCols + 56];
945             }
946             if (i < nData - nCols + 57) {
947                 nPixels[65] = P[i + nCols + 57];
948             }
949             if (i < nData - nCols + 58) {
950                 nPixels[66] = P[i + nCols + 58];
951             }
952             if (i < nData - nCols + 59) {
953                 nPixels[67] = P[i + nCols + 59];
954             }
955             if (i < nData - nCols + 60) {
956                 nPixels[68] = P[i + nCols + 60];
957             }
958             if (i < nData - nCols + 61) {
959                 nPixels[69] = P[i + nCols + 61];
960             }
961             if (i < nData - nCols + 62) {
962                 nPixels[70] = P[i + nCols + 62];
963             }
964             if (i < nData - nCols + 63) {
965                 nPixels[71] = P[i + nCols + 63];
966             }
967             if (i < nData - nCols + 64) {
968                 nPixels[72] = P[i + nCols + 64];
969             }
970             if (i < nData - nCols + 65) {
971                 nPixels[73] = P[i + nCols + 65];
972             }
973             if (i < nData - nCols + 66) {
974                 nPixels[74] = P[i + nCols + 66];
975             }
976             if (i < nData - nCols + 67) {
977                 nPixels[75] = P[i + nCols + 67];
978             }
979             if (i < nData - nCols + 68) {
980                 nPixels[76] = P[i + nCols + 68];
981             }
982             if (i < nData - nCols + 69) {
983                 nPixels[77] = P[i + nCols + 69];
984             }
985             if (i < nData - nCols + 70) {
986                 nPixels[78] = P[i + nCols + 70];
987             }
988             if (i < nData - nCols + 71) {
989                 nPixels[79] = P[i + nCols + 71];
990             }
991             if (i < nData - nCols + 72) {
992                 nPixels[80] = P[i + nCols + 72];
993             }
994             if (i < nData - nCols + 73) {
995                 nPixels[81] = P[i + nCols + 73];
996             }
997             if (i < nData - nCols + 74) {
998                 nPixels[82] = P[i + nCols + 74];
999             }
999             if (i < nData - nCols + 75) {
1000                nPixels[83] = P[i + nCols + 75];
1001            }
1002            if (i < nData - nCols + 76) {
1003                nPixels[84] = P[i + nCols + 76];
1004            }
1005            if (i < nData - nCols + 77) {
1006                nPixels[85] = P[i + nCols + 77];
1007            }
1008            if (i < nData - nCols + 78) {
1009                nPixels[86] = P[i + nCols + 78];
1010            }
1011            if (i < nData - nCols + 79) {
1012                nPixels[87] = P[i + nCols + 79];
1013            }
1014            if (i < nData - nCols + 80) {
1015                nPixels[88] = P[i + nCols + 80];
1016            }
1017            if (i < nData - nCols + 81) {
1018                nPixels[89] = P[i + nCols + 81];
1019            }
1020            if (i < nData - nCols + 82) {
1021                nPixels[90] = P[i + nCols + 82];
1022            }
1023            if (i < nData - nCols + 83) {
1024                nPixels[91] = P[i + nCols + 83];
1025            }
1026            if (i < nData - nCols + 84) {
1027                nPixels[92] = P[i + nCols + 84];
1028            }
1029            if (i < nData - nCols + 85) {
1030                nPixels[93] = P[i + nCols + 85];
1031            }
1032            if (i < nData - nCols + 86) {
1033                nPixels[94] = P[i + nCols + 86];
1034            }
1035            if (i < nData - nCols + 87) {
1036                nPixels[95] = P[i + nCols + 87];
1037            }
1038            if (i < nData - nCols + 88) {
1039                nPixels[96] = P[i + nCols + 88];
1040            }
1041            if (i < nData - nCols + 89) {
1042                nPixels[97] = P[i + nCols + 89];
1043            }
1044            if (i < nData - nCols + 90) {
1045                nPixels[98] = P[i + nCols + 90];
1046            }
1047            if (i < nData - nCols + 91) {
1048                nPixels[99] = P[i + nCols + 91];
1049            }
1050            if (i < nData - nCols + 92) {
1051                nPixels[100] = P[i + nCols + 92];
1052            }
1053            if (i < nData - nCols + 93) {
1054                nPixels[101] = P[i + nCols + 93];
1055            }
1056            if (i < nData - nCols + 94) {
1057                nPixels[102] = P[i + nCols + 94];
1058            }
1059            if (i < nData - nCols + 95) {
1060                nPixels[103] = P[i + nCols + 95];
1061            }
1062            if (i < nData - nCols + 96) {
1063                nPixels[104] = P[i + nCols + 96];
1064            }
1065            if (i < nData - nCols + 97) {
1066                nPixels[105] = P[i + nCols + 97];
1067            }
1068            if (i < nData - nCols + 98) {
1069                nPixels[106] = P[i + nCols + 98];
1070            }
1071            if (i < nData - nCols + 99) {
1072                nPixels[107] = P[i + nCols + 99];
1073            }
1074            if (i < nData - nCols + 100) {
1075                nPixels[108] = P[i + nCols + 100];
1076            }
1077            if (i < nData - nCols + 101) {
1078                nPixels[109] = P[i + nCols + 101];
1079            }
1080            if (i < nData - nCols + 102) {
1081                nPixels[110] = P[i + nCols + 102];
1082            }
1083            if (i < nData - nCols + 103) {
1084                nPixels[111] = P[i + nCols + 103];
1085            }
1086            if (i < nData - nCols + 104) {
1087                nPixels[112] = P[i + nCols + 104];
1088            }
1089            if (i < nData - nCols + 105) {
1090                nPixels[113] = P[i + nCols + 105];
1091            }
1092            if (i < nData - nCols + 106) {
1093                nPixels[114] = P[i + nCols + 106];
1094            }
1095            if (i < nData - nCols + 107) {
1096                nPixels[115] = P[i + nCols + 107];
1097            }
1098            if (i < nData - nCols + 108) {
1099                nPixels[116] = P[i + nCols + 108];
1100            }
1101            if (i < nData - nCols + 109) {
1102                nPixels[117] = P[i + nCols + 109];
1103            }
1104            if (i < nData - nCols + 110) {
1105                nPixels[118] = P[i + nCols + 110];
1106            }
1107            if (i < nData - nCols + 111) {
1108                nPixels[119] = P[i + nCols + 111];
1109            }
1110            if (i < nData - nCols + 112) {
1111                nPixels[120] = P[i + nCols + 112];
1112            }
1113            if (i < nData - nCols + 113) {
1114                nPixels[121] = P[i + nCols + 113];
1115            }
1116            if (i < nData - nCols + 114) {
1117                nPixels[122] = P[i + nCols + 114];
1118            }
1119            if (i < nData - nCols + 115) {
1120                nPixels[123] = P[i + nCols + 115];
1121            }
1122            if (i < nData - nCols + 116) {
1123                nPixels[124] = P[i + nCols + 116];
1124            }
1125            if (i < nData - nCols + 117) {
1126                nPixels[125] = P[i + nCols + 117];
1127            }
1128            if (i < nData - nCols + 118) {
1129                nPixels[126] = P[i + nCols + 118];
1130            }
1131            if (i < nData - nCols + 119) {
1132                nPixels[127] = P[i + nCols + 119];
1133            }
1134            if (i < nData - nCols + 120) {
1135                nPixels[128] = P[i + nCols + 120];
1136            }
1137            if (i < nData - nCols + 121) {
1138                nPixels[129] = P[i + nCols + 121];
1139            }
1140            if (i < nData - nCols + 122) {
1141                nPixels[130] = P[i + nCols + 122];
1142            }
1143            if (i < nData - nCols + 123) {
1144                nPixels[131] = P[i + nCols + 123];
1145            }
1146            if (i < nData - nCols + 124) {
1147                nPixels[132] = P[i + nCols + 124];
1148            }
1149            if (i < nData - nCols + 125) {
1150                nPixels[133] = P[i + nCols + 125];
1151            }
1152            if (i < nData - nCols + 126) {
1153                nPixels[134] = P[i + nCols + 126];
1154            }
1155            if (i < nData - nCols + 127) {
1156                nPixels[135] = P[i + nCols + 127];
1157            }
1158            if (i < nData - nCols + 128) {
1159                nPixels[136] = P[i + nCols + 128];
1160            }
1161            if (i < nData - nCols + 129) {
1162                nPixels[137] = P[i + nCols + 129];
1163            }
1164            if (i < nData - nCols + 130) {
1165                nPixels[138] = P[i + nCols + 130];
1166            }
1167            if (i < nData - nCols + 131) {
1168                nPixels[139] = P[i + nCols + 131];
1169            }
1170            if (i < nData - nCols + 132) {
1171                nPixels[140] = P[i + nCols + 132];
1172            }
1173            if (i < nData - nCols + 133) {
1174                nPixels[141] = P[i + nCols + 133];
1175            }
1176            if (i < nData - nCols + 134) {
1177                nPixels[142] = P[i + nCols + 134];
1178            }
1179            if (i < nData - nCols + 135) {
1180                nPixels[143] = P[i + nCols + 135];
1181            }
1182            if (i < nData - nCols + 136) {
1183                nPixels[144] = P[i + nCols + 136];
1184            }
1185            if (i < nData - nCols + 137) {
1186                nPixels[145] = P[i + nCols + 137];
1187            }
1188            if (i < nData - nCols + 138) {
1189                nPixels[146] = P[i + nCols + 138];
1190            }
1191            if (i < nData - nCols + 139) {
1192                nPixels[147] = P[i + nCols + 139];
1193            }
1194            if (i < nData - nCols + 140) {
1195                nPixels[148] = P[i + nCols + 140];
1196            }
1197            if (i < nData - nCols + 141) {
1198                nPixels[149] = P[i + nCols + 141];
1199            }
1199        }
1200    }
1201
1202    // Set the first pixel
1203    if (0[0] == 0) {
1204        P[0] = 0;
1205    } else if (0[0] == 1) {
1206        P[0] = 1;
1207    } else {
1208        throw Exception::PixelValueOutOfBoundsException();
1209    }
1210
1211    // Walk through the toprow and determine if it's a new
1212    // blob or it's connected
1213    // with previously determine blob
1214    for (uint32_t i = 1; i < nCols; i++) {
1215        if (0[i] == 0) {
1216            P[i] = 0;
1217        } else if (0[i] == 1) {
1218            // If West is zero assume this is a new blob
1219            if (P[i - 1] == 0) {
1220                P[i] = ++currentlbl;
1221                vector<uint16_t> cVector;
1222                cVector.push_back(currentlbl);
1223                adj.push_back(cVector);
1224            } else { // set as previous blob
1225                P[i] = P[i - 1];
1226            }
1227        } else { // Value of of bounds
1228            throw Exception::PixelValueOutOfBoundsException();
1229        }
1230    }
1231
1232    // walk through each pixel and determine if it's a new
1233    // blob or it's connected
1234    // with previously determine blob
1235    for (uint32_t i = OriginalImg.cols; i < nData; i++) {
1236        if (0[i] == 0) { // Original pixel = 0
1237            P[i] = 0;
1238        } else if (0[i] == 1) {
1239            // Get an array of Neighboring Pixels
1240            uint16_t *nPixels = new uint16_t[noConn[nRow[i]]];
1241            if (nRow[i] != 1) {
1242                nPixels[0] = P[i - 1];
1243            }
1244            if (i < nData - nCols) {
1245                nPixels[1] = P[i + 1];
1246            }
1247            if (i > nData - nCols - 1) {
1248                nPixels[2] = P[i - nCols - 1];
1249            }
1250            if (i > nData - nCols - 2) {
1251                nPixels[3] = P[i - nCols];
1252            }
1253            if (i < nData - nCols - 2) {
1254                nPixels[4] = P[i + nCols];
1255            }
1256            if (i < nData - nCols - 3) {
1257                nPixels[5] = P[i + nCols - 1];
1258            }
1259            if (i > nData - nCols - 3) {
1260                nPixels[6] = P[i - nCols - 2];
1261            }
1262            if (i > nData - nCols - 4) {
1263                nPixels[7] = P[i - nCols - 3];
1264            }
1265            if (i < nData - nCols + 1) {
1266                nPixels[8] = P[i + nCols + 1];
1267            }
1268            if (i < nData - nCols + 2) {
1269                nPixels[9] = P[i + nCols + 2];
1270            }
1271            if (i < nData - nCols + 3) {
1272                nPixels[10] = P[i + nCols + 3];
1273            }
1274            if (i < nData - nCols + 4) {
1275                nPixels[11] = P[i + nCols + 4];
1276            }
1277            if (i < nData - nCols + 5) {
1278                nPixels[12] = P[i + nCols + 5];
1279            }
1280            if (i < nData - nCols + 6) {
1281                nPixels[13] = P[i + nCols + 6];
1282            }
1283            if (i < nData - nCols + 7) {
1284                nPixels[14] = P[i + nCols + 7];
1285            }
1286            if (i < nData - nCols + 8) {
1287                nPixels[15] = P[i + nCols + 8];
1288            }
1289            if (i < nData - nCols + 9) {
1290                nPixels[16] = P[i + nCols + 9];
1291            }
1292            if (i < nData - nCols + 10) {
1293                nPixels[17] = P[i + nCols + 10];
1294            }
1295            if (i < nData - nCols + 11) {
1296                nPixels[18] = P[i + nCols + 11];
1297            }
1298            if (i < nData - nCols + 12) {
1299                nPixels[19] = P[i + nCols + 12];
1300            }
1301            if (i < nData - nCols + 13) {
1302                nPixels[20] = P[i + nCols + 13];
1303            }
1304            if (i < nData - nCols + 14) {
1305                nPixels[21] = P[i + nCols + 14];
1306            }
1307            if (i < nData - nCols + 15) {
1308                nPixels[22] = P[i + nCols + 15];
1309            }
1310            if (i < nData - nCols + 16) {
1311                nPixels[23] = P[i + nCols + 16];
1312            }
1313            if (i < nData - nCols + 17) {
1314                nPixels[24] = P[i + nCols + 17];
1315            }
1316            if (i < nData - nCols + 18) {
1317                nPixels[25] = P[i + nCols + 18];
1318            }
1319            if (i < nData - nCols + 19) {
1320                nPixels[26] = P[i + nCols + 19];
1321            }
1322            if (i < nData - nCols + 20) {
1323                nPixels[27] = P[i + nCols + 20];
1324            }
1325            if (i < nData - nCols + 21) {
1326                nPixels[28] = P[i + nCols + 21];
1327            }
1328            if (i < nData - nCols + 22) {
1329                nPixels[29] = P[i + nCols + 22];
1330            }
1331            if (i < nData - nCols + 23) {
1332                nPixels[30] = P[i + nCols + 23];
1333            }
1334            if (i < nData - nCols + 24) {
1335                nPixels[31] = P[i + nCols + 24];
1336            }
1337            if (i < nData - nCols + 25) {
1338                nPixels[32] = P[i + nCols + 25];
1339            }
1339        }
1340    }
1341
1342    // Set the first pixel
1343    if (0[0] == 0) {
1344        P[0] = 0;
1345    } else if (0[0] == 1) {
1346        P[0] = 1;
1347    } else {
1348        throw Exception::PixelValueOutOfBoundsException();
1349    }
1350
1351    // Walk through the toprow and determine if it's a new
1352    // blob or it's connected
1353    // with previously determine blob
1354    for (uint32_t i = 1; i < nCols; i++) {
1355        if (0[i] == 0) {
1356            P[i] = 0;
1357        } else if (0[i] == 1) {
1358            // If West is zero assume this is a new blob
1359            if (P[i - 1] == 0) {
1360                P[i] = ++currentlbl;
1361                vector<uint16_t> cVector;
1362                cVector.push_back(currentlbl);
1363                adj.push_back(cVector);
1364            } else { // set as previous blob
1365                P[i] = P[i - 1];
1366            }
1367        } else { // Value of of bounds
1368            throw Exception::PixelValueOutOfBoundsException();
1369        }
1370    }
1371
1372    // walk through each pixel and determine if it's a new
1373    // blob or it's connected
1374    // with previously determine blob
1375    for (uint32_t i = OriginalImg.cols; i < nData; i++) {
1376        if (0[i] == 0) { // Original pixel = 0
1377            P[i] = 0;
1378        } else if (0[i] == 1) {
1379            // Get an array of Neighboring Pixels
1380            uint16_t *nPixels = new uint16_t[noConn[nRow[i]]];
1381            if (nRow[i] != 1) {
1382                nPixels[0] = P[i - 1];
1383            }
1384            if (i < nData - nCols) {
1385                nPixels[1] = P[i + 1];
1386            }
1387            if (i > nData - nCols - 1) {
1388                nPixels[2] = P[i - nCols - 1];
1389            }
1390            if (i > nData - nCols - 2) {
1391                nPixels[3] = P[i - nCols];
1392            }
1393            if (i < nData - nCols - 2) {
1394                nPixels[4] = P[i + nCols];
1395            }
1396            if (i < nData - nCols - 3) {
1397                nPixels[5] = P[i + nCols - 1];
1398            }
1399            if (i > nData - nCols - 3) {
1400                nPixels[6] = P[i - nCols - 2];
1401            }
1402            if (i > nData - nCols - 4) {
1403                nPixels[7] = P[i - nCols - 3];
1404            }
1405            if (i < nData - nCols + 1) {
1406                nPixels[8] = P[i + nCols + 1];
1407            }
1408            if (i < nData - nCols + 2) {
1409                nPixels[9] = P[i + nCols + 2];
1410            }
1411            if (i < nData - nCols + 3) {
1412                nPixels[10] = P[i + nCols + 3];
1413            }
1414            if (i < nData - nCols + 4) {
1415                nPixels[11] = P[i + nCols + 4];
1416            }
1417            if (i < nData - nCols + 5) {
1418                nPixels[12] = P[i + nCols + 5];
1419            }
1420            if (i < nData - nCols + 6) {
1421                nPixels[13] = P[i + nCols + 6];
1422            }
1423            if (i < nData - nCols + 7) {
1424                nPixels[14] = P[i + nCols + 7];
1425            }
1426            if (i < nData - nCols + 8) {
1427                nPixels[15] = P[i + nCols + 8];
1428            }
1429            if (i < nData - nCols + 9) {
1430                nPixels[16] = P[i + nCols + 9];
1431            }
1432            if (i < nData - nCols + 10) {
1433                nPixels[17] = P[i + nCols + 10];
1434            }
1435            if (i < nData - nCols + 11) {
1436                nPixels[18] = P[i + nCols + 11];
1437            }
1438            if (i < nData - nCols + 12) {
1439                nPixels[19] = P[i + nCols + 12];
1440            }
1441            if (i < nData - nCols + 13) {
1442                nPixels[20] = P[i + nCols + 13];
1443            }
1444            if (i < nData - nCols + 14) {
1445                nPixels[21] = P[i + nCols + 14];
1446            }
1447            if (i < nData - nCols + 15) {
1448                nPixels[22] = P[i + nCols + 15];
1449            }
1450            if (i < nData - nCols + 16) {
1451                nPixels[23] = P[i + nCols + 16];
1452            }
1453            if (i < nData - nCols + 17) {
1454                nPixels[24] = P[i + nCols + 17];
1455            }
1456            if (i < nData - nCols + 18) {
1457                nPixels[25] = P[i + nCols + 18];
1458            }
1459            if (i < nData - nCols + 19) {
1460                nPixels[26] = P[i + nCols + 19];
1461            }
1462            if (i < nData - nCols + 20) {
1463                nPixels[27] = P[i + nCols + 20];
1464            }
1465            if (i < nData - nCols + 21) {
1466                nPixels[28] = P[i + nCols + 21];
1467            }
1468            if (i < nData - nCols + 22) {
1469                nPixels[29] = P[i + nCols + 22];
1470            }
1471            if (i < nData - nCols + 23) {
1472                nPixels[30] = P[i + nCols + 23];
1473            }
1474            if (i < nData - nCols + 24) {
1475                nPixels[31] = P[i + nCols + 24];
1476            }
1477            if (i < nData - nCols + 25) {
1478                nPixels[32] = P[i + nCols + 25];
1479            }
1479        }
1480    }
1481
1482    // Set the first pixel
1483    if (0[0] == 0) {
1484        P[0] = 0;
1485    } else if (0[0] == 1) {
1486        P[0]
```

```

750     }
751     uint32_t j = i - nCols - ((nRow[i] == 1) ? 0 : ((conn
752         == Four) ? 0 : 1));
753     for_each(nPixels + ((nRow[i] != 1) ? 1 : 0), nPixels +
754         noConn[nRow[i]],
755         [&](uint16_t &N) { N = P[j++]; });
756
757     // Sort the neighbors for easier checking
758     SoilMath::Sort::QuickSort<uint16_t>(nPixels, noConn[
759         nRow[i]]);
760
761     // If all are zero assume this is a new blob
762     if (nPixels[lastConn[nRow[i]]] == 0) {
763         P[i] = ++currentlbl;
764         vector<uint16_t> cVector;
765         cVector.push_back(currentlbl);
766         adj.push_back(cVector);
767     } else {
768         /* Sets the processed value to the smallest non-zero
769            value and update
770            * the connectedLabels */
771         for (uint32_t j = 0; j < noConn[nRow[i]]; j++) {
772             if (nPixels[j] > 0) {
773                 P[i] = nPixels[j];
774                 break;
775             }
776         }
777         /* If previous blobs belong to different connected
778            components set the
779            * current processed value to the lowest value and
780            remember that the
781            * other values should be the lowest value*/
782         if (P[i] != nPixels[lastConn[nRow[i]]]) {
783             for (int j = lastConn[nRow[i]]; j >= 0; --j) {
784                 if (nPixels[j] <= P[i]) {
785                     break;
786                 } else {
787                     adj[nPixels[j]].push_back(P[i]);
788                 }
789             }
790         }
791     }
792     delete[] nPixels;
793 }
794
795 /*!
796 * \brief Segment::InvertAdjacencyList invert the
797 * adjacencylist for upstream
798 * (unused)
799 * \param adj

```

```

799 * \param adjInv
800 */
801 void Segment::InvertAdjacencyList(std::vector<std::vector<
802     uint16_t>> &adj,
803                                     std::vector<std::vector<
804     uint16_t>> &adjInv) {
805
806     // Build the inverted vector
807     adjInv.resize(adj.size());
808     uint16_t count = 0;
809     for_each(adj.begin(), adj.end(), [&](std::vector<uint16_t>
810             &V) {
811         for_each(V.begin(), V.end(),
812                 [&](uint16_t &C) { adjInv[C].push_back(count);
813             });
814         count++;
815     });
816 }
817
818 /*!
819 * \brief Segment::MakeConsecutive make the valueArr
820 * consecutive numbers
821 * \param valueArr
822 * \param noElem
823 * \param maxLabel
824 */
825 void Segment::MakeConsecutive(uint16_t *valueArr, uint32_t
826     noElem,
827                                     uint16_t &maxLabel) {
828     std::vector<std::vector<uint16_t>> conseq;
829     conseq.resize(noElem);
830     for (uint32_t i = 0; i < noElem; i++) {
831         conseq[valueArr[i]].push_back(i);
832     }
833     uint32_t count = 1;
834     for (uint32_t i = 1; i < noElem; i++) {
835         if (conseq[i].size() > 0) {
836             for (uint32_t j = 0; j < conseq[i].size(); j++) {
837                 valueArr[conseq[i][j]] = count;
838             }
839             count++;
840         }
841     }
842     maxLabel = count - 1;
843 }
844
845 /*!
846 * \brief Segment::MakeConsecutive probably a fault in this
847 * function. Don't use
848 * \param valueArr
849 * \param keyArr
850 * \param noElem
851 * \param maxlabel
852 */
853 void Segment::MakeConsecutive(uint16_t *valueArr, uint16_t *
854     keyArr,
855                                     uint16_t noElem, uint16_t &
856     maxlabel);

```

```
                                maxlabel) {
847     SoilMath::Sort::QuickSort<uint16_t>(valueArr, keyArr,
848     noElem);
849     uint16_t count = 0;
850     for (uint32_t i = 1; i < noElem; i++) {
851         if (valueArr[i] != valueArr[i - 1]) {
852             count++;
853             valueArr[i] = count;
854         }
855     SoilMath::Sort::QuickSort<uint16_t>(keyArr, valueArr,
856     noElem);
857     delete [] keyArr;
858     maxlabel = count;
859 }
```

H.0.21 General project files

```

1 #-----
2 #
3 # Project created by QtCreator 2015-06-06T12:07:42
4 #
5 #-----
6
7 QT      += core concurrent
8 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
9 QMAKE_CXXFLAGS += -std=c++11
10
11 TARGET = SoilVision
12 TEMPLATE = lib
13 VERSION = 0.9.2
14
15 DEFINES += SOILVISION_LIBRARY
16 unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../../../build/install/
17
18 SOURCES += \
19     Segment.cpp \
20     MorphologicalFilter.cpp \
21     ImageProcessing.cpp \
22     Enhance.cpp \
23     Conversion.cpp
24
25 HEADERS += \
26     WrongKernelSizeException.h \
27     VisionDebug.h \
28     Vision.h \
29     Segment.h \
30     PixelValueOutOfBoundsException.h \
31     MorphologicalFilter.h \
32     ImageProcessing.h \
33     Enhance.h \
34     EmptyImageException.h \
35     ConversionNotSupportedException.h \
36     Conversion.h \
37     ChannelMismatchException.h
38
39 unix {
40     target.path = $$PWD/../../../../../build/install
41     INSTALLS += target
42 }
43
44 #opencv
45 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui - \
46         opencv_imgproc
46 INCLUDEPATH += /usr/local/include/opencv
47 INCLUDEPATH += /usr/local/include
48
49 #boost
50 DEFINES += BOOST_ALL_DYN_LINK
51 INCLUDEPATH += /usr/include/boost
52
53 unix:!macx: LIBS += -L$$PWD/../../../../../build/install/ -lSoilMath

```

```

54
55 INCLUDEPATH += $$PWD/../SoilMath
56 DEPENDPATH += $$PWD/../SoilMath


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11 #include "Conversion.h"
12 #include "Enhance.h"
13 #include "Segment.h"
14 #include "MorphologicalFilter.h"


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11 // Debuging helper macros
12 #ifndef DEBUG
13 // #define DEBUG
14 #endif
15
16 #ifdef DEBUG
17 #include <limits>
18 #include <opencv2/highgui/highgui.hpp>
19 #include <vector>
20 #include "ImageProcessing.h"
21 #ifndef SHOW_DEBUG_IMG
22 #define SHOW_DEBUG_IMG(img, T1, maxVal, windowName, scale)
23   \
24   Vision::ImageProcessing::ShowDebugImg<T1>(img, maxVal,
25   windowName, scale)
26 #endif // !SHOW_DEBUG_IMG
27 #else
28 #ifndef SHOW_DEBUG_IMG
29 #define SHOW_DEBUG_IMG(img, T1, maxVal, windowName, scale)
30 #endif // !SHOW_DEBUG_IMG
31 #endif


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited

```

```
3  * and only allowed with the written consent of the author (Jelle Spijker)
4  * This software is proprietary and confidential
5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6  */
7
8 /*! \class ChannelMismatchException
9 Exception class which is thrown when Extracted channel out
10 of bounds exception
11 */
12 #pragma once
13
14 #include <exception>
15 #include <string>
16
17 using namespace std;
18
19 namespace Vision {
20 namespace Exception {
21 class ChannelMismatchException : public std::exception {
22 public:
23     ChannelMismatchException(
24         string m = "Extracted channel out of bounds exception!
25         ")
26     : msg(m){};
27     ~ChannelMismatchException() __GLIBCXX_USE_NOEXCEPT{};
28     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
29         msg.c_str(); };
30
31     private:
32     string msg;
33 };
34
35 /* Copyright (C) Jelle Spijker - All Rights Reserved
36  * Unauthorized copying of this file, via any medium is
37  * strictly prohibited
38  * and only allowed with the written consent of the author (Jelle Spijker)
39  * This software is proprietary and confidential
40  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
41  */
42
43 /*! \class ConversionNotSupportedException
44 Exception class which is thrown when an illegal conversion
45 is requested.
46 */
47 #pragma once
48
49 #include <exception>
50 #include <string>
51
52 using namespace std;
53
```

```

18 namespace Vision {
19 namespace Exception {
20 class ConversionNotSupportedException : public std::
21 exception {
21 public:
22     ConversionNotSupportedException(
23         string m = "Requested conversion is not supported!")
24         : msg(m){};
25     ~ConversionNotSupportedException() __GLIBCXX_USE_NOEXCEPT
26     {};
26     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
27         msg.c_str(); };
27
28 private:
29     string msg;
30 };
31 }
32 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
8 /*! \class EmtpyImageException
9 Exception class which is thrown when operations are about to
10 start on a empty
11 image.
11 */
12
13 #pragma once
14
15 #include <exception>
16 #include <string>
17
18 using namespace std;
19
20 namespace Vision {
21 namespace Exception {
22 class EmtpyImageException : public std::exception {
23 public:
24     EmtpyImageException(string m = "Empty Image!") : msg(m){};
25     ~EmtpyImageException() __GLIBCXX_USE_NOEXCEPT{};
26     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
27         msg.c_str(); };
27
28 private:
29     string msg;
30 };
31 }
32 }

```

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10 /*! \class PixelValueOutOfBoundsException
11 Exception class which is thrown when an unexpected pixel
12 value has to be
13 computed
14 */
15 #pragma once
16
17 #include <exception>
18 #include <string>
19
20 using namespace std;
21
22 namespace Vision {
23 namespace Exception {
24 class PixelValueOutOfBoundsException : public std::exception
25 {
26 public:
27     PixelValueOutOfBoundsException(string m = "Current pixel
28         value out of bounds!")
29     : msg(m){};
30     ~PixelValueOutOfBoundsException() __GLIBCXX_USE_NOEXCEPT{};
31     const char *what() const __GLIBCXX_USE_NOEXCEPT { return
32         msg.c_str(); };
33
34 private:
35     string msg;
36 };
37 }
38 }
```

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10 /*! \class WrongKernelSizeException
11 Exception class which is thrown when a wrong kernelsize is
12 requested
13 */
14 #pragma once
15
16 #include <exception>
```

```
14 #include <string>
15
16 using namespace std;
17
18 namespace Vision {
19 namespace Exception {
20 class WrongKernelSizeException : public std::exception {
21 public:
22     WrongKernelSizeException(string m = "Wrong kernel
23         dimensions!") : msg(m){};
24     ~WrongKernelSizeException() _GLIBCXX_USE_NOEXCEPT{};
25     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
26         msg.c_str(); };
27     private:
28     string msg;
29 }
30 }
```



I. Analyzer Library

I.0.22 Analyzer Class

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11 #define STARTING_ESTIMATE_PROGRESS 300
12 #ifndef DEBUG
13 //#define DEBUG
14 #endif
15
16 #include <opencv2/core.hpp>
17 #include <opencv2/imgproc.hpp>
18 #include <vector>
19 #include <cmath>
20
21 #include "sample.h"
22 #include "soilsettings.h"
23 #include "soilanalyzerexception.h"
24
25 #include "SoilMath.h"
26
27 #include <QtCore/QObject>
28 #include <QThread>
29 #include <QtConcurrent>
30
31 #include "Vision.h"
```

```

30
31 namespace SoilAnalyzer {
32 class Analyzer : public QObject {
33     Q_OBJECT
34
35 public:
36     bool PredictShape = true;
37     float CurrentSIfactor = 0.0111915;
38     bool SIfactorDet = false;
39     struct Image_t {
40         cv::Mat FrontLight;
41         cv::Mat BackLight;
42         float SIPixelFactor = 0.0111915;
43     }; /*!< */
44
45     typedef std::vector<Image_t> Images_t; /*!< */
46     Images_t *Snapshots = nullptr;           /*!< */
47     SoilSettings *Settings = nullptr;         /*!< */
48
49     Sample *Results; /*!< */
50
51     Analyzer(Images_t *snapshots, Sample *results,
52               SoilSettings *settings);
53
54     void Analyse();
55     void Analyse(Images_t *snapshots, Sample *results,
56                  SoilSettings *settings);
57     float CalibrateSI(float SI, cv::Mat &img);
58
59     uint32_t MaxProgress = STARTING_ESTIMATE_PROGRESS; /*!< */
60
61     SoilMath::NN NeuralNet; /*!< */
62
63 signals:
64     void on_progressUpdate(int value); /*!< */
65     void on_maxProgressUpdate(int value); /*!< */
66     void on_AnalysisFinished(); /*!< */
67
68 private:
69     uint32_t currentProgress = 0; /*!< */
70     uint32_t currentParticleID = 0; /*!< */
71     double BinRanges[15]{0.0, 0.038, 0.045, 0.063, 0.075,
72                          0.09, 0.125, 0.18,
73                          0.25, 0.355, 0.5, 0.71, 1.0, 1.4,
74                          2.0};
75
76     SoilMath::FFT fft; /*!< */
77
78     void CalcMaxProgress();
79     void CalcMaxProgressAnalyze();
80     void PrepImages();
81     void GetBW(std::vector<cv::Mat> &images, std::vector<cv::
82                Mat> &BWvector);
83     void GetBW(cv::Mat &img, cv::Mat &BW);
84
85     void GetEnhancedInt(Images_t *snapshots,

```

```

81             std::vector<cv::Mat> &intensityVector)
82             ;
83     void GetEnhancedInt(cv::Mat &img, cv::Mat &intensity);
84     void GetParticles(std::vector<cv::Mat> &BW, Images_t *
85                         snapshots,
86                         Particle::ParticleVector_t &
87                         partPopulation);
86     void GetParticlesFromBlobList(Vision::Segment::BlobList_t
87                         &bloblist,
88                         Image_t *snapshot,
89                         Particle::ParticleVector_t &
90                         partPopulation);
90     void CleanUpMatVector(std::vector<cv::Mat> &mv);
91     void CleanUpMatVector(Images_t *mv);
92     void GetFFD(Particle::ParticleVector_t &particalPopulation
93                  );
94     void GetPrediction(Particle::ParticleVector_t &
95                         particlePopulation);
96 };
97 }



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8 */
9
8 #include "analyzer.h"
9
10 namespace SoilAnalyzer {
11
12 /*!
13  *\brief Analyzer::Analyzer
14  *\param snapshots
15  *\param results
16  *\param settings
17 */
18 Analyzer::Analyzer(Images_t *snapshots, Sample *results,
19                     SoilSettings *settings = nullptr) {
20     this->Snapshots = snapshots;
21     this->Results = results;
22     if (settings == nullptr) {
23         Settings = new SoilSettings;
24     } else {
25         this->Settings = settings;
26     }
27     NeuralNet.LoadState(Settings->NNlocation);
28 }
29

```



```

81         ucharStat_t(Results->GetRoundnessVector()->data(),
82                         Results->GetRoundnessVector()->size(), 1,
83                         5, 0, true);
84     Results->PSD =
85         SoilMath::PSD(Results->GetPSDVector()->data(),
86                         Results->GetPSDVector()->size(),
87                         BinRanges, 15, 14);
88     emit on_progressUpdate(currentProgress++);
89     emit on_AnalysisFinished();
90 }
91
92 void Analyzer::CleanUpMatVector(std::vector<Mat> &mv) {
93     for_each(mv.begin(), mv.end(), [](cv::Mat &I) { I.release
94         (); });
95     mv.clear();
96 }
97 /**
98  * \brief Analyzer::CleanUpMatVector
99  * \param mv
100 */
101 void Analyzer::CleanUpMatVector(Images_t *mv) {
102     for_each(mv->begin(), mv->end(), [](Image_t &I) {
103         I.BackLight.release();
104         I.FrontLight.release();
105     });
106     mv->clear();
107 }
108 /**
109  * \brief Analyzer::CalcMaxProgress
110 */
111 void Analyzer::CalcMaxProgress() {
112     // Static processing steps
113     MaxProgress += Snapshots->size() * 5;
114
115     // Optional processing steps
116     if (Settings->useBlur) {
117         MaxProgress += Snapshots->size();
118     }
119     if (Settings->useAdaptiveContrast) {
120         MaxProgress += Snapshots->size();
121     }
122     if (Settings->fillHoles) {
123         MaxProgress += Snapshots->size();
124     }
125     if (Settings->ignorePartialBorderParticles) {
126         MaxProgress += Snapshots->size();
127     }
128     if (Settings->morphFilterType != Vision::
129         MorphologicalFilter::NONE) {
130         MaxProgress += Snapshots->size();
131     }
132 }

```

```

133     emit on_maxProgressUpdate(MaxProgress);
134 }
135
136 void Analyzer::CalcMaxProgressAnalyze() {
137     MaxProgress -= STARTING_ESTIMATE_PROGRESS;
138     MaxProgress += Results->ParticlePopulation.size() * 2;
139
140     emit on_maxProgressUpdate(MaxProgress);
141 }
142
143 /**
144  * \brief Analyzer::GetEnhancedInt
145  * \param snapshots
146  * \param intensityVector
147  */
148 void Analyzer::GetEnhancedInt(Images_t *snapshots,
149                               std::vector<Mat> &
150                               intensityVector) {
151     if (Settings->useBacklightProjection) {
152         for_each(snapshots->begin(), snapshots->end(), [&]{
153             Image_t &I {
154                 cv::Mat intensity;
155                 GetEnhancedInt(I.BackLight, intensity);
156                 intensityVector.push_back(intensity);
157             });
158     } else {
159         for_each(snapshots->begin(), snapshots->end(), [&]{
160             Image_t &I {
161                 cv::Mat intensity;
162                 GetEnhancedInt(I.FrontLight, intensity);
163                 intensityVector.push_back(intensity);
164             });
165     }
166 /**
167  * \brief Analyzer::GetEnhancedInt
168  * \param img
169  * \param intensity
170  */
171 void Analyzer::GetEnhancedInt(Mat &img, Mat &intensity) {
172     Vision::Conversion IntConvertor(img.clone());
173     IntConvertor.Convert(Vision::Conversion::RGB, Vision::
174         Conversion::Intensity);
175     emit on_progressUpdate(currentProgress++);
176     SHOW_DEBUG_IMG(IntConvertor.ProcessedImg, uchar, 255, "RGB
177         2 Int", false);
178
179     if (Settings->useBlur) {
180         Vision::Enhance IntBlur(IntConvertor.ProcessedImg.clone
181             ());
182         IntBlur.Blur(Settings->blurKernelSize);
183         emit on_progressUpdate(currentProgress++);
184         uint32_t HBK = Settings->blurKernelSize / 2;
185         uint32_t BK = Settings->blurKernelSize - 1;
186         if (Settings->useAdaptiveContrast) {

```

```

183     Vision::Enhance IntAdaptContrast(
184         IntBlur.ProcessedImg(
185             cv::Rect(HBK, HBK, IntBlur.
186                 ProcessedImg.cols - BK,
187                 IntBlur.ProcessedImg.rows -
188                 BK)).clone());
189     IntAdaptContrast.AdaptiveContrastStretch(
190         Settings->adaptContrastKernelSize,
191         Settings->adaptContrastKernelFactor);
192     emit on_progressUpdate(currentProgress++);
193     uint32_t HAK = Settings->adaptContrastKernelSize / 2;
194     uint32_t AK = Settings->adaptContrastKernelSize - 1;
195     intensity = IntAdaptContrast.ProcessedImg(
196         cv::Rect(HAK, HAK, IntAdaptContrast.ProcessedImg.
197             cols - AK,
198                 IntAdaptContrast.ProcessedImg.rows - AK));
199
200 } else {
201     intensity = IntBlur.ProcessedImg(
202         cv::Rect(HBK, HBK, IntBlur.ProcessedImg.cols - BK,
203                 IntBlur.ProcessedImg.rows - BK));
204 }
205 } else if (Settings->useAdaptiveContrast) {
206     Vision::Enhance IntAdaptContrast(IntConvertor.
207         ProcessedImg.clone());
208     IntAdaptContrast.AdaptiveContrastStretch(
209         Settings->adaptContrastKernelSize, Settings->
210             adaptContrastKernelFactor);
211     emit on_progressUpdate(currentProgress++);
212     uint32_t HAK = Settings->adaptContrastKernelSize / 2;
213     uint32_t AK = Settings->adaptContrastKernelSize - 1;
214     intensity = IntAdaptContrast.ProcessedImg(
215         cv::Rect(HAK, HAK, IntAdaptContrast.ProcessedImg.
216             cols - AK,
217                 IntAdaptContrast.ProcessedImg.rows - AK));
218 } else {
219     intensity = IntConvertor.ProcessedImg;
220 }
221 SHOW_DEBUG_IMG(intensity, uchar, 255, "Enhanced Int",
222     false);
223 }
224 */
225 * \brief Analyzer::GetBW
226 * \param images
227 * \param BWvector
228 */
229 void Analyzer::GetBW(std::vector<cv::Mat> &images,
230                     std::vector<cv::Mat> &BWvector) {
231     for_each(images.begin(), images.end(), [&](cv::Mat &I) {
232         cv::Mat BW;
233         GetBW(I, BW);
234         BWvector.push_back(BW);
235     });
236 }

```

```
231  /*!
232  * \brief Analyzer::GetBW
233  * \param img
234  * \param BW
235  */
236 void Analyzer::GetBW(cv::Mat &img, cv::Mat &BW) {
237     Vision::Segment SegBL(img.clone());
238     SegBL.sigma = Settings->sigmaFactor;
239     SegBL.thresholdOffset = Settings->thresholdOffsetValue;
240     SegBL.ConvertToBW(Settings->typeOfObjectsSegmented);
241     emit on_progressUpdate(currentProgress++);
242     SHOW_DEBUG_IMG(SegBL.ProcessedImg, uchar, 255, "Segment",
243                     true);
244
245     cv::Mat BWholes;
246     if (Settings->fillHoles) {
247         Vision::Segment Fillholes(SegBL.ProcessedImg);
248         Fillholes.FillHoles();
249         BWholes = Fillholes.ProcessedImg;
250         emit on_progressUpdate(currentProgress++);
251         SHOW_DEBUG_IMG(BWholes, uchar, 255, "Fillholes", true);
252     } else {
253         BWholes = SegBL.ProcessedImg;
254     }
255
256     cv::Mat BWborder;
257     if (Settings->ignorePartialBorderParticles) {
258         Vision::Segment RemoveBB(BWholes.clone());
259         RemoveBB.RemoveBorderBlobs();
260         BWborder = RemoveBB.ProcessedImg;
261         emit on_progressUpdate(currentProgress++);
262         SHOW_DEBUG_IMG(BWborder, uchar, 255, "RemoveBorderBlobs",
263                         true);
263     } else {
264         BWborder = BWholes;
265     }
266
267     if (Settings->morphFilterType != Vision::
268         MorphologicalFilter::NONE) {
269         Vision::MorphologicalFilter Morph(BWborder.clone());
270         cv::Mat kernel = cv::Mat::zeros(Settings->filterMaskSize
271                                         ,
272                                         Settings->filterMaskSize
273                                         ,
274                                         CV_8UC1);
275         uint32_t hMaskSize = Settings->filterMaskSize / 2;
276         cv::circle(kernel, cv::Point(hMaskSize, hMaskSize),
277                    hMaskSize + 1, 1, -1);
278         switch (Settings->morphFilterType) {
279             case Vision::MorphologicalFilter::CLOSE:
280                 Morph.Close(kernel);
281                 break;
282             case Vision::MorphologicalFilter::OPEN:
283                 Morph.Open(kernel);
284                 break;
285             case Vision::MorphologicalFilter::DILATE:
286                 Morph.Dilation(kernel);
287         }
288     }
289 }
```

```

281     break;
282     case Vision::MorphologicalFilter::ERODE:
283         Morph.Erosion(kernel);
284         break;
285     case Vision::MorphologicalFilter::NONE:
286         Morph.ProcessedImg = Morph.OriginalImg;
287         break;
288     }
289     BW = Morph.ProcessedImg;
290     emit on_progressUpdate(currentProgress++);
291     SHOW_DEBUG_IMG(BW, uchar, 255, "Morphological operation"
292         , true);
293 } else {
294     BW = BWholes;
295 }
296 }
297 */
298 * \brief Analyzer::GetParticles
299 * \param BW
300 * \param snapshots
301 * \param partPopulation
302 */
303 void Analyzer::GetParticles(std::vector<Mat> &BW, Images_t *
304     snapshots,
305     Particle::ParticleVector_t &
306     partPopulation) {
307     for (uint32_t i = 0; i < snapshots->size(); i++) {
308         Vision::Segment prepBW(BW[i]);
309         prepBW.GetBlobList();
310         emit on_progressUpdate(currentProgress++);
311         GetParticlesFromBlobList(prepBW.BlobList, &(snapshots->
312             at(i)),
313             partPopulation);
314         emit on_progressUpdate(currentProgress++);
315     }
316 }
317 */
318 * \brief Analyzer::GetParticlesFromBlobList
319 * \param bloblist
320 * \param snapshot
321 * \param edge
322 * \param partPopulation
323 */
324 void Analyzer::GetParticlesFromBlobList(
325     Vision::Segment::BlobList_t &bloblist, Image_t *snapshot
326     ,
327     Particle::ParticleVector_t &partPopulation) {
328     for_each(bloblist.begin(), bloblist.end(), [&](Vision::
329         Segment::Blob_t &B) {
330         Particle part;
331         part.ID = currentParticleID++;
332         part.PixelArea = B.Area;
333         Vision::Segment::getOriententated(B.Img, B.Centroid, B.
334             Theta,

```

```

330                     part.Eccentricity);
331     cv::Mat RGB = Vision::Segment::CopyMat<uchar>(snapshot->
332                     FrontLight(B.ROI),
333                     B.Img,
334                     CV_8UC3
335                     ).clone
336                     ());
337     cv::Rect ROI;
338     Vision::Segment::RotateImg(B.Img, part.BW, B.Theta, B.
339                     Centroid, ROI);
340     Vision::Segment::RotateImg(RGB, part.RGB, B.Theta, B.
341                     Centroid, ROI);
342     Vision::Segment edgeSeg(part.BW);
343     edgeSeg.GetEdgesEroding();
344     part.Edge = edgeSeg.ProcessedImg.clone();
345     part.SIPixelFactor = snapshot->SIPixelFactor;
346     part.isPreparedForAnalysis = false;
347     part.SetRoundness();
348     partPopulation.push_back(part);
349   });
350 }
351
352 /*!
353  * \brief Analyzer::GetFFD
354  * \param particlePopulation
355  */
356 void Analyzer::GetFFD(Particle::ParticleVector_t &
357   particlePopulation) {
358   //for_each(particlePopulation.begin(), particlePopulation.
359   //           end(), [&](Particle &P) {
360   QtConcurrent::blockingMap<Particle::ParticleVector_t>(
361     particlePopulation, [&](Particle &P) {
362       if (!P.isPreparedForAnalysis) {
363         try {
364           SoilMath::FFT fft;
365           P.FFDescriptors = fft.GetDescriptors(P.Edge);
366           P.isPreparedForAnalysis = true;
367         } catch (SoilMath::Exception::MathException &e) {
368           if (*e.id() == EXCEPTION_NO_CONTOUR_FOUND_NR) {
369             P.isSmall = true;
370           }
371         }
372         emit on_progressUpdate(currentProgress++);
373       }
374     });
375   });
376 }
377
378 /*!
379  * \brief Analyzer::GetPrediction
380  * \param particlePopulation
381  */
382 void Analyzer::GetPrediction(Particle::ParticleVector_t &
383   particlePopulation) {
384   for_each(particlePopulation.begin(), particlePopulation.
385           end(),
386           [&](Particle &P) {

```

```

376     if (P.isPreparedForAnalysis) {
377         if (!P.isSmall) {
378             ComplexVect_t usedFFDescr(P.FFDescriptors.
379                                         begin(),
380                                         P.FFDescriptors.
381                                         begin() +
382                                         NeuralNet.
383                                         GetInputNeurons
384                                         ());
385             P.Classification = NeuralNet.Predict(
386                                         usedFFDescr);
387             P.isAnalysed = true;
388         }
389     }
390 }
391
392 float Analyzer::CalibrateSI(float SI, Mat &img) {
393     Vision::Conversion greyConv(img);
394     greyConv.Convert(Vision::Conversion::RGB, Vision::
395                     Conversion::Intensity);
396     Vision::Segment segment(greyConv.ProcessedImg);
397     segment.ConvertToBW(Vision::Segment::Dark);
398     segment.GetBlobList(true);
399     uint32_t maxCircle = 0;
400     for_each(segment.BlobList.begin(), segment.BlobList.end(),
401               [&](Vision::Segment::Blob_t &B) {
402                   if (B.ROI.height > maxCircle) {
403                       maxCircle = B.ROI.height;
404                   }
405                   if (B.ROI.width > maxCircle) {
406                       maxCircle = B.ROI.width;
407                   }
408               });
409     qDebug() << "Maximum circle in pixels: " << maxCircle;
410     CurrentSIfactor = SI / maxCircle;
411     qDebug() << "Current SI factor : " << CurrentSIfactor;
412     return CurrentSIfactor;
413 }
414

```

I.0.23 Sample Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include "stdint.h"
13 #include <vector>
14 #include <string>
15 #include "Stats.h"
16 #include "psd.h"
17 #include "particle.h"
18 #include <fstream>
19 #include <boost/archive/binary_iarchive.hpp>
20 #include <boost/archive/binary_oarchive.hpp>
21 #include <boost/serialization/string.hpp>
22 #include <boost/serialization/version.hpp>
23 #include <boost/serialization/vector.hpp>
24 #include <boost/iostreams/filter/zlib.hpp>
25 #include <boost/iostreams/filtering_streambuf.hpp>
26 #include "zlib.h"
27 #include "soilanalyzertypes.h"
28
29 namespace SoilAnalyzer {
30 class Sample {
31 public:
32     Sample();
33     uint32_t ID;           /*!< The sample ID*/
34     std::string Location; /*!< The Location where the sample
35     was taken*/
36     double Longitude = 4.629618299999947;
37     double Latitude = 51.8849149;
38     double Depth = 0;
39     std::string Date = "01-09-2015";
40     std::string Name; /*!< The sample name identifier*/
41     Particle::ParticleVector_t
42         ParticlePopulation; /*!< the individual particles of
43         the sample*/
44     SoilMath::PSD PSD; /*!< The Particle Size Distribution*/
45     ucharStat_t Roundness;
46     ucharStat_t Angularity;
47     floatStat_t RI; /*!< The statistical Redness Index data*/
48     void Save(const std::string &filename);
49     void Load(const std::string &filename);
50 }
```

```

51     Particle::PSDVector_t *GetPSDVector();
52     Particle::ClassVector_t *GetRoundnessVector();
53     Particle::ClassVector_t *GetAngularityVector();
54     Particle::doubleVector_t *GetCIELab_aVector();
55     Particle::doubleVector_t *GetCIELab_bVector();
56
57     bool isPreparedForAnalysis =
58         false; /*!< is the sample ready for analysis, are all
59             the particles
60                 extracted*/
61     bool isAnalysed = false; /*!< is the sample analyzed*/
62
63     bool ChangesSinceLastSave = false;
64     bool ParticleChangedStatePSD = false;
65     bool ParticleChangedStateClass = false;
66     bool ParticleChangedStateRoundness = false;
67     bool ParticleChangedStateAngularity = false;
68     bool ColorChange = false;
69
70     bool IsLoadedFromDisk = false;
71
72 private:
73     Particle::PSDVector_t Diameter; /*!< The PSD raw data*/
74     bool PSDGathered = false; /*!< is the raw data
75             gathered*/
76     Particle::ClassVector_t RoundnessVec;
77     bool RoundnessGathered = false;
78     Particle::ClassVector_t AngularityVec;
79     bool AngularityGathered = false;
80     Particle::doubleVector_t CIELab_aVec;
81     bool CIELab_aGathered = false;
82     Particle::doubleVector_t CIELab_bVec;
83     bool CIELab_bGathered = false;
84
85     friend class boost::serialization::access;
86     template <class Archive>
87     void serialize(Archive &ar, const unsigned int version) {
88         ar &ID;
89         ar &Location;
90         ar &Name;
91         ar &ParticlePopulation;
92         ar &Diameter;
93         ar &RoundnessVec;
94         ar &AngularityVec;
95         ar &PSD;
96         ar &Roundness;
97         ar &Angularity;
98         ar &RI;
99         ar &isPreparedForAnalysis;
100        ar &isAnalysed;
101        ar &ChangesSinceLastSave;
102        ar &ParticleChangedStatePSD;
103        ar &ParticleChangedStateClass;
104        ar &ParticleChangedStateAngularity;
105        ar &ParticleChangedStateRoundness;
106        ar &PSDGathered;

```

```

105     ar &RoundnessGathered;
106     ar &AngularityGathered;
107     ar &IsLoadedFromDisk;
108     if (version > 0) {
109         ar &Longitude;
110         ar &Latitude;
111         ar &Date;
112         ar &Depth;
113         ar &AngularityVec;
114         ar &AngularityGathered;
115         ar &CIELab_aVec;
116         ar &CIELab_aGathered;
117         ar &CIELab_bVec;
118         ar &CIELab_bGathered;
119         ar &ColorChange;
120     } else {
121         Latitude = 51.8849149;
122         Longitude = 4.629618299999947;
123         Date = "01-10-2015";
124         Depth = 0;
125         CIELab_aVec = Particle::doubleVector_t();
126         CIELab_aGathered = false;
127         CIELab_bVec = Particle::doubleVector_t();
128         CIELab_bGathered = false;
129         ColorChange = false;
130     }
131 }
132 };
133 }
134 BOOST_CLASS_VERSION(SoilAnalyzer::Sample, 1)

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10
11  namespace SoilAnalyzer {
12  namespace io = boost::iostreams;
13
14  /**
15   * \brief Sample::Sample
16   */
17  Sample::Sample() {}
18
19  /**
20   * \brief Sample::Save
21   * \param filename
22   */
23  void Sample::Save(const std::string &filename) {

```

```

24     std::ofstream ofs(filename.c_str(), std::ios::out | std::
25         ios::binary);
26     {
27         io::filtering_streambuf<io::output> out;
28         out.push(io::zlib_compressor(io::zlib::best_compression)
29             );
30         out.push(ofs);
31         {
32             boost::archive::binary_oarchive oa(out);
33             oa << boost::serialization::make_nvp("Sample", *this);
34         }
35         ofs.close();
36     }
37
38 /*!
39 * \brief Sample::Load
40 * \param filename
41 */
42 void Sample::Load(const std::string &filename) {
43     std::ifstream ifs(filename.c_str(), std::ios::in | std::
44         ios::binary);
45     {
46         io::filtering_streambuf<io::input> in;
47         in.push(io::zlib_decompressor());
48         in.push(ifs);
49         {
50             boost::archive::binary_iarchive ia(in);
51             ia >> boost::serialization::make_nvp("Sample", *this);
52         }
53     }
54     ifs.close();
55 }
56
57 /*!
58 * \brief Sample::GetPSDVector
59 * \return
60 */
61 Particle::PSDVector_t *Sample::GetPSDVector() {
62     if (!PSDGathered || ParticleChangedStatePSD) {
63         Diameter.clear();
64         for_each(ParticlePopulation.begin(), ParticlePopulation.
65             end(),
66             [&](Particle &P) { Diameter.push_back(P.
67                 GetSiDiameter()); });
68     PSDGathered = true;
69     ParticleChangedStatePSD = false;
70 }
71
72 Particle::ClassVector_t *Sample::GetAngularityVector() {
73     if (!AngularityGathered || ParticleChangedStateAngularity)
74     {

```

```
74     AngularityVec.clear();
75     for_each(ParticlePopulation.begin(), ParticlePopulation.
76             end(),
77             [&](Particle &P) { AngularityVec.push_back(P.
78                 GetAngularity()); });
79     AngularityGathered = true;
80     ParticleChangedStateAngularity = false;
81 }
82
83 Particle::ClassVector_t *Sample::GetRoundnessVector() {
84     if (!RoundnessGathered || ParticleChangedStateRoundness) {
85         RoundnessVec.clear();
86         for_each(ParticlePopulation.begin(), ParticlePopulation.
87             end(),
88             [&](Particle &P) { RoundnessVec.push_back(P.
89                 GetRoundness()); });
90     RoundnessGathered = true;
91     ParticleChangedStateRoundness = false;
92 }
93
94 Particle::doubleVector_t *Sample::GetCIELab_aVector() {
95     if (!CIELab_aGathered || ColorChange) {
96         CIELab_aVec.clear();
97         for_each(ParticlePopulation.begin(), ParticlePopulation.
98             end(),
99             [&](Particle &P) { CIELab_aVec.push_back(P.
100                 getMeanLab().a); });
101     CIELab_aGathered = true;
102 }
103
104 Particle::doubleVector_t *Sample::GetCIELab_bVector() {
105     if (!CIELab_bGathered || ColorChange) {
106         CIELab_bVec.clear();
107         for_each(ParticlePopulation.begin(), ParticlePopulation.
108             end(),
109             [&](Particle &P) { CIELab_bVec.push_back(P.
110                 getMeanLab().b); });
111     CIELab_bGathered = true;
112 }
113 }
```

I.0.24 Particle Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9  #pragma once
10
11 #include <opencv2/core.hpp>
12 #include <stdint.h>
13 #include <vector>
14 #include "SoilMath.h"
15 #include <fstream>
16 #include <boost/archive/binary_iarchive.hpp>
17 #include <boost/archive/binary_oarchive.hpp>
18 #include <boost/serialization/string.hpp>
19 #include <boost/serialization/version.hpp>
20 #include <boost/serialization/vector.hpp>
21 #include <boost/iostreams/filter/zlib.hpp>
22 #include <boost/iostreams/filtering_streambuf.hpp>
23 #include "zlib.h"
24 #include "soilanalyzerexception.h"
25 #include "lab_t_archive.h"
26 #include "soilanalyzertypes.h"
27 #include "Vision.h"
28
29 namespace SoilAnalyzer {
30 class Particle {
31 public:
32     typedef std::vector<Particle>
33     ParticleVector_t; /*!< a vector consisting of
34     individual particles*/
35     typedef std::vector<double> PSDVector_t; /*!< a vector
36     used in the PSD*/
37     typedef std::vector<uint8_t>
38     ClassVector_t; /*!< a vector used in the
39     classification histogram*/
40     typedef std::vector<float> floatVector_t;
41     typedef std::vector<double> doubleVector_t;
42     Particle();
43     uint32_t ID; /*!< The particle ID*/
44     cv::Mat BW; /*!< The binary image of the particle*/
45     cv::Mat Edge; /*!< The binary edge image of the particle*/
46     cv::Mat RGB; /*!< The RGB image of the particle*/
47     Point_t Centroid = {0, 0};
48     std::vector<Complex_t> FFDescriptors; /*!< The Fast
49     Fourier Descriptors

```

```
48                                     describing the
49                                     contour in the
50                                     Frequency domain
51                                     */
50     Predict_t Classification;           /*!< The
51     classification prediction*/
51     double SIPixelFactor = 0.0111915; /*!< The conversion
52     factor from pixel to SI*/
52     uint32_t PixelArea = 0;           /*!< The total area of
53     the binary image*/
53     double Eccentricity = 1;
54
55     float GetSIVolume();
56     float GetSiDiameter();
57     uint8_t GetRoundness();
58     uint8_t GetAngularity();
59     float GetMeanRI();
60     Lab_t getMeanLab();
61
62     void SetRoundness();
63
64     void Save(const std::string &filename);
65     void Load(const std::string &filename);
66
67     bool isPreparedForAnalysis = false; /*!< is the particle
68     ready for analysis*/
68     bool isAnalysed = false;           /*!< is the particle
69     analyzed*/
69     bool isSmall = false;
70
71 private:
72     float SIVolume = 0.; /*!< The correspondening SI volume*/
73     float SIDiameter = 0.;
74
75     float meanRI = 0;
76     Lab_t meanLab{0,0,0};
77     cv::Mat LAB;
78
79     void getLabImg();
80
81     friend class boost::serialization::access;
82     template <class Archive>
83     void serialize(Archive &ar, const unsigned int version) {
84
85         ar &ID;
86         ar &BW;
87         ar &Edge;
88         ar &RGB;
89         ar &FFDescriptors;
90         ar &Classification;
91         ar &SIPixelFactor;
92         ar &PixelArea;
93         ar &SIVolume;
94         ar &isPreparedForAnalysis;
95         ar &isAnalysed;
96         if (version > 0) {
```

```

97     ar &isSmall;
98     ar &SIDiameter;
99     ar &Centroid.x;
100    ar &Centroid.y;
101    ar &Eccentricity;
102 } else {
103     isSmall = false;
104     SIDiameter = GetSiDiameter();
105     Centroid.x = 0;
106     Centroid.y = 0;
107     Eccentricity = 1;
108 }
109 if (version > 1) {
110     ar &meanLab;
111     ar &meanRI;
112 }
113 else {
114     meanLab.L = 0;
115     meanLab.a = 0;
116     meanLab.b = 0;
117 }
118 }
119 };
120 }
121 BOOST_CLASS_VERSION(SoilAnalyzer::Particle, 2)

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8  */
9
8 #include "particle.h"
9
10 namespace SoilAnalyzer {
11 namespace io = boost::iostreams;
12
13 Particle::Particle() {}
14
15 /*!
16  * \brief Particle::Save
17  * \param filename
18  */
19 void Particle::Save(const std::string &filename) {
20     std::ofstream ofs(filename.c_str(), std::ios::out | std::ios::binary);
21     {
22         io::filtering_streambuf<io::output> out;
23
24         out.push(io::zlib_compressor(io::zlib::best_compression)
25                 );
26         out.push(ofs);
27     }

```

```
27         boost::archive::binary_oarchive oa(out);
28         oa << boost::serialization::make_nvp("Particle", *this
29             );
30     }
31     ofs.close();
32 }
33
34 /*
35  * \brief Particle::Load
36  * \param filename
37  */
38 void Particle::Load(const std::string &filename) {
39     std::ifstream ifs(filename.c_str(), std::ios::in | std::ios::binary);
40     {
41         io::filtering_streambuf<io::input> in;
42
43         in.push(io::zlib_decompressor());
44         in.push(ifs);
45     {
46         boost::archive::binary_iarchive ia(in);
47         ia >> boost::serialization::make_nvp("Particle", *this
48             );
49     }
50     ifs.close();
51 }
52
53 /*
54  * \brief Particle::GetSIVolume
55  * \return
56  */
57 float Particle::GetSIVolume() {
58     if (SIVolume == 0.) {
59         if (PixelArea == 0) {
60             throw Exception::SoilAnalyzerException(
61                 EXCEPTION_PARTICLE_NOT_ANALYZED,
62                 EXCEPTION_PARTICLE_NOT_ANALYZED_NR);
63         }
64         SIVolume = SoilMath::calcVolume(PixelArea) *
65             SIPixelFactor * (Eccentricity/2 + 0.5);
66     }
67     return SIVolume;
68 }
69
70 float Particle::GetSiDiameter() {
71     if (SIDiameter == 0.) {
72         if (PixelArea == 0) {
73             throw Exception::SoilAnalyzerException(
74                 EXCEPTION_PARTICLE_NOT_ANALYZED,
75                 EXCEPTION_PARTICLE_NOT_ANALYZED_NR);
76         }
77         SIDiameter = SoilMath::calcDiameter(PixelArea) *
78             SIPixelFactor * (Eccentricity/2 + 0.5);
79     }
80 }
```

```

76     return SIDiameter;
77 }
78
79 uint8_t Particle::GetAngularity() {
80     uint8_t angularity = ((Classification.Category - 1) % 6) +
81     1;
82     return angularity;
83 }
84
85 uint8_t Particle::GetRoundness() {
86     uint8_t roundness = ((Classification.Category - 1) / 6) +
87     1;
88     return roundness;
89 }
90
91 void Particle::SetRoundness() {
92     uint8_t ang = GetAngularity() - 1;
93     Classification.Category +=
94         ang + (static_cast<uint8_t>(floor(Eccentricity / 0.33)) *
95         6);
96     Classification.ManualSet = true;
97 }
98
99 Lab_t Particle::getMeanLab() {
100    if (BW.empty() || RGB.empty()) {
101        throw SoilAnalyzer::Exception::SoilAnalyzerException(
102            EXCEPTION_NO_IMAGES_PRESENT,
103            EXCEPTION_NO_IMAGES_PRESENT_NR);
104    }
105    if (meanLab.L == 0 && meanLab.a == 0 && meanLab.b == 0) {
106        // convert to Lab
107        if (LAB.empty()) {
108            getLabImg();
109        }
110        std::vector<cv::Mat> LABvect = Vision::Conversion::
111            extractChannel(LAB);
112        std::vector<float> labvect;
113        for_each(LABvect.begin(), LABvect.end(), [&](cv::Mat &I)
114        {
115            floatStat_t labStat((float *)I.data, I.rows, I.cols, (
116                uchar *)BW.data, 1,
117                0, true);
118            labvect.push_back(labStat.Mean);
119        });
120        meanLab.L = labvect[0];
121        meanLab.a = labvect[1];
122        meanLab.b = labvect[2];
123    }
124    return meanLab;
125 }
126
127 float Particle::GetMeanRI() {
128    if (BW.empty() || RGB.empty()) {
129        throw SoilAnalyzer::Exception::SoilAnalyzerException(
130            EXCEPTION_NO_IMAGES_PRESENT,
131            EXCEPTION_NO_IMAGES_PRESENT_NR);

```

```
124     }
125     if (meanRI == 0) {
126         if (LAB.empty()) {
127             getLabImg();
128         }
129         Vision::Conversion convertor(LAB);
130         convertor.Convert(Vision::Conversion::CIE_lab, Vision::
131             Conversion::RI);
131         floatStat_t RIstat((float *)convertor.ProcessedImg.data,
132             LAB.rows, LAB.cols,
133             (uchar *)BW.data, 1, 0, true);
133         meanRI = RIstat.Mean;
134     }
135     return meanRI;
136 }
137
138 void Particle::getLabImg() {
139     Vision::Conversion convertor(RGB);
140     convertor.Convert(Vision::Conversion::RGB, Vision::
141         Conversion::CIE_lab);
141     LAB = convertor.ProcessedImg.clone();
142 }
143 }
```

```

46                                     contrast
47                                     stretch*/
48     bool useBlur = false; /*< Should the mediaan blur be used
49     during analysis*/
50     uint32_t blurKernelSize = 5; /*< the median blurkernel*/
51
52     Vision::Segment::TypeOfObjects typeOfObjectsSegmented =
53         Vision::Segment::Dark; /*< Which type of object
54         should be segmented*/
55     bool ignorePartialBorderParticles =
56         true; /*< Indication of partial border particles
57         should be used*/
58     bool fillHoles = true; /*< should the holes be filled*/
59     float sigmaFactor = 2; /*< The sigma factor or the
60         bandwidth indicating which
61             pixel intensity values count
62                 belong to an object*/
63
64     int thresholdOffsetValue = 0; /*< an tweaking offset
65         value*/
66
67     Vision::MorphologicalFilter::FilterType morphFilterType =
68         Vision::MorphologicalFilter::OPEN; /*< Indicating
69             which type of
70                 morphological
71                     filter should
72                         be
73                             used*/
74
75     uint32_t filterMaskSize = 5; /*< the filter
76         mask*/
77
78     uint32_t HDRframes =
79         5; /*< The number of frames which should be used for
80             the HDR image*/
81     float lightLevel = 0.5; /*< The light level of the
82         environmental case*/
83     bool encInv = false; /*< invert the values gained form
84         the encoder*/
85     bool enableRainbow =
86         true; /*< run a rainbow loop on the RGB encoder
87             during analysis*/
88     bool useBacklightProjection = true; /*<!< use
89         Projection*/
90     bool useHDR = false; /*<!< use HDR
91         */
92     std::string defaultWebcam = "USB Microscope"; /*<!< The
93         defaultWebcam string*/
94     int Brightness_front = 0; /*<!< cam brightness setting
95         front light*/
96     int Brightness_proj = -10; /*<!< cam brightness setting
97         projected light*/
98     int Contrast_front = 36; /*<!< cam contrast setting front
99         light*/
100    int Contrast_proj = 36; /*<!< cam contrast setting
101        projected light*/

```

```

79     int Saturation_front = 64; /*!< cam saturation setting
80     front light*/
81     int Saturation_proj = 0;    /*!< cam saturation setting
82     projected light*/
83     int Hue_front = 0;          /*!< cam hue setting front
84     light*/
85     int Hue_proj = -40;         /*!< cam hue setting projected
86     light*/
87     int Gamma_front = 100;      /*!< cam gamma setting front
88     light*/
89     int Gamma_proj = 200;        /*!< cam gamma setting
90     projected light*/
91     int PowerLineFrequency_front =
92     1; /*!< cam powerline freq setting front light*/
93     int PowerLineFrequency_proj =
94     1; /*!< cam powerline freq setting
95     projected light*/
96     int Sharpness_front = 12;    /*!< cam sharpness setting front
97     light*/
98     int Sharpness_proj = 25;     /*!< cam sharpness setting
99     projected light*/
100    int BackLightCompensation_front =
101    1; /*!< cam backlight compensation setting front light
102    */
103    int BackLightCompensation_proj =
104    1; /*!< cam backlight compensation setting projected
105    light*/
106    std::string NNlocation = "NeuralNet/Default.NN";
107    bool useCUDA = false; /*!< CUDA enabled*/
108    int selectedResolution = 0;
109    std::string SampleFolder = "~/Samples";
110    std::string SettingsFolder = "Settings";
111    std::string NNFolder = "NeuralNet";
112    std::string StandardSentTo = "j.spijker@ihcmerwede.com";
113    std::string StandardPrinter = "PDF printer";
114    uint32_t StandardNumberOfShots = 10;
115    bool PredictTheShape = true;
116    bool Revolution = true;
117    private:
118    friend class boost::serialization::access;
119    template <class Archive>
120    void serialize(Archive &ar, const unsigned int version) {
121        if (version >= 0) {
122            ar &BOOST_SERIALIZATION_NVP(useAdaptiveContrast);
123            ar &BOOST_SERIALIZATION_NVP(adaptContrastKernelFactor)
124            ;
125            ar &BOOST_SERIALIZATION_NVP(adaptContrastKernelSize);
126            ar &BOOST_SERIALIZATION_NVP(useBlur);
127            ar &BOOST_SERIALIZATION_NVP(blurKernelSize);
128            ar &BOOST_SERIALIZATION_NVP(typeOfObjectsSegmented);
129            ar &BOOST_SERIALIZATION_NVP(
130                ignorePartialBorderParticles);
131            ar &BOOST_SERIALIZATION_NVP(fillHoles);
132            ar &BOOST_SERIALIZATION_NVP(sigmaFactor);
133            ar &BOOST_SERIALIZATION_NVP(morphFilterType);
134            ar &BOOST_SERIALIZATION_NVP(filterMaskSize);

```

```

122     ar &BOOST_SERIALIZATION_NVP(thresholdOffsetValue);
123     ar &BOOST_SERIALIZATION_NVP(HDRframes);
124     ar &BOOST_SERIALIZATION_NVP(lightLevel);
125     ar &BOOST_SERIALIZATION_NVP(encInv);
126     ar &BOOST_SERIALIZATION_NVP(enableRainbow);
127     ar &BOOST_SERIALIZATION_NVP(useBacklightProjection);
128     ar &BOOST_SERIALIZATION_NVP(useHDR);
129     ar &BOOST_SERIALIZATION_NVP(defaultWebcam);
130     ar &BOOST_SERIALIZATION_NVP(Brightness_front);
131     ar &BOOST_SERIALIZATION_NVP(Brightness_proj);
132     ar &BOOST_SERIALIZATION_NVP(Contrast_front);
133     ar &BOOST_SERIALIZATION_NVP(Contrast_proj);
134     ar &BOOST_SERIALIZATION_NVP(Saturation_front);
135     ar &BOOST_SERIALIZATION_NVP(Saturation_proj);
136     ar &BOOST_SERIALIZATION_NVP(Hue_front);
137     ar &BOOST_SERIALIZATION_NVP(Hue_proj);
138     ar &BOOST_SERIALIZATION_NVP(Gamma_front);
139     ar &BOOST_SERIALIZATION_NVP(Gamma_proj);
140     ar &BOOST_SERIALIZATION_NVP(PowerLineFrequency_front);
141     ar &BOOST_SERIALIZATION_NVP(PowerLineFrequency_proj);
142     ar &BOOST_SERIALIZATION_NVP(Sharpness_front);
143     ar &BOOST_SERIALIZATION_NVP(Sharpness_proj);
144     ar &BOOST_SERIALIZATION_NVP(
145         BackLightCompensation_front);
146     ar &BOOST_SERIALIZATION_NVP(BackLightCompensation_proj
147         );
148     ar &BOOST_SERIALIZATION_NVP(NNlocation);
149     ar &BOOST_SERIALIZATION_NVP(useCUDA);
150     ar &BOOST_SERIALIZATION_NVP(selectedResolution);
151     ar &BOOST_SERIALIZATION_NVP(SampleFolder);
152     ar &BOOST_SERIALIZATION_NVP(SettingsFolder);
153     ar &BOOST_SERIALIZATION_NVP(NNFolder);
154     ar &BOOST_SERIALIZATION_NVP(StandardSentTo);
155     ar &BOOST_SERIALIZATION_NVP(StandardPrinter);
156     ar &BOOST_SERIALIZATION_NVP(StandardNumberOfShots);
157 }
158 }
159 };
160 }
161 BOOST_CLASS_VERSION(SoilAnalyzer::SoilSettings, 0)

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #include "soilsettings.h"
11
12 namespace SoilAnalyzer {
13     SoilSettings::SoilSettings() {}
14 }

```

```
12
13 void SoilSettings::LoadSettings(string filename) {
14     std::ifstream ifs(filename.c_str());
15     boost::archive::xml_iarchive ia(ifs);
16     ia >> boost::serialization::make_nvp("SoilSettings", *this
17     );
18 }
19 void SoilSettings::SaveSettings(string filename) {
20     std::ofstream ofs(filename.c_str());
21     boost::archive::xml_oarchive oa(ofs);
22     oa << boost::serialization::make_nvp("SoilSettings", *this
23     );
24 }
```

I.0.26 General project files

```
1 #-----
2 #
3 # Project created by QtCreator 2015-08-08T18:57:27
4 #
5 #-----
6
7 QT      += core gui concurrent
8 QMAKE_CXXFLAGS += -std=c++11
9
10 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
11 @@
12 CONFIG(release, debug|release):DEFINES += QT_NO_DEBUG_OUTPUT
13 @@
14
15 TARGET = SoilAnalyzer
16 TEMPLATE = lib
17 VERSION = 0.9.96
18
19 DEFINES += SOILANALYZER_LIBRARY
20
21 SOURCES += \
22     soilsettings.cpp \
23     sample.cpp \
24     particle.cpp \
25     analyzer.cpp
26
27 HEADERS += \
28     soilsettings.h \
29     sample.h \
30     particle.h \
31     analyzer.h \
32     soilanalyzerexception.h \
33     soilanalyzer.h \
34     lab_t_archive.h \
35     soilanalyzertypes.h
36
37 #opencv
38 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui
39 INCLUDEPATH += /usr/local/include/opencv
40 INCLUDEPATH += /usr/local/include
41
42 #boost
43 DEFINES += BOOST_ALL_DYN_LINK
44 INCLUDEPATH += /usr/include/boost
45 LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_serialization -lboost_iostreams
46
47 #Zlib
48 LIBS += -L/usr/local/lib -lz
49 INCLUDEPATH += /usr/local/include
50
51 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
52 INCLUDEPATH += $$PWD/../SoilMath
53 DEPENDPATH += $$PWD/../SoilMath
```

```

54
55 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
56   lSoilVision
56 INCLUDEPATH += $$PWD/./SoilVision
57 DEPENDPATH += $$PWD/./SoilVision
58
59 #MainLib
60
61 target.path = $$PWD/../../build/install
62 INSTALLS += target

```

```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
9
8 #pragma once
10
10 #include <boost/archive/binary_iarchive.hpp>
11 #include <boost/archive/binary_oarchive.hpp>
12 #include <boost/serialization/access.hpp>
13 #include "soilanalyzertypes.h"
14
15 namespace boost {
16 namespace serialization {
17 /*!
18  * \brief serialize Serialize the openCV mat to disk
19 */
20 template <class Archive>
21 inline void serialize(Archive &ar, SoilAnalyzer::Lab_t &P,
22   const unsigned int version __attribute__((unused))) {
23   ar &P.L;
24   ar &P.a;
25   ar &P.b;
26 }
27 }

```

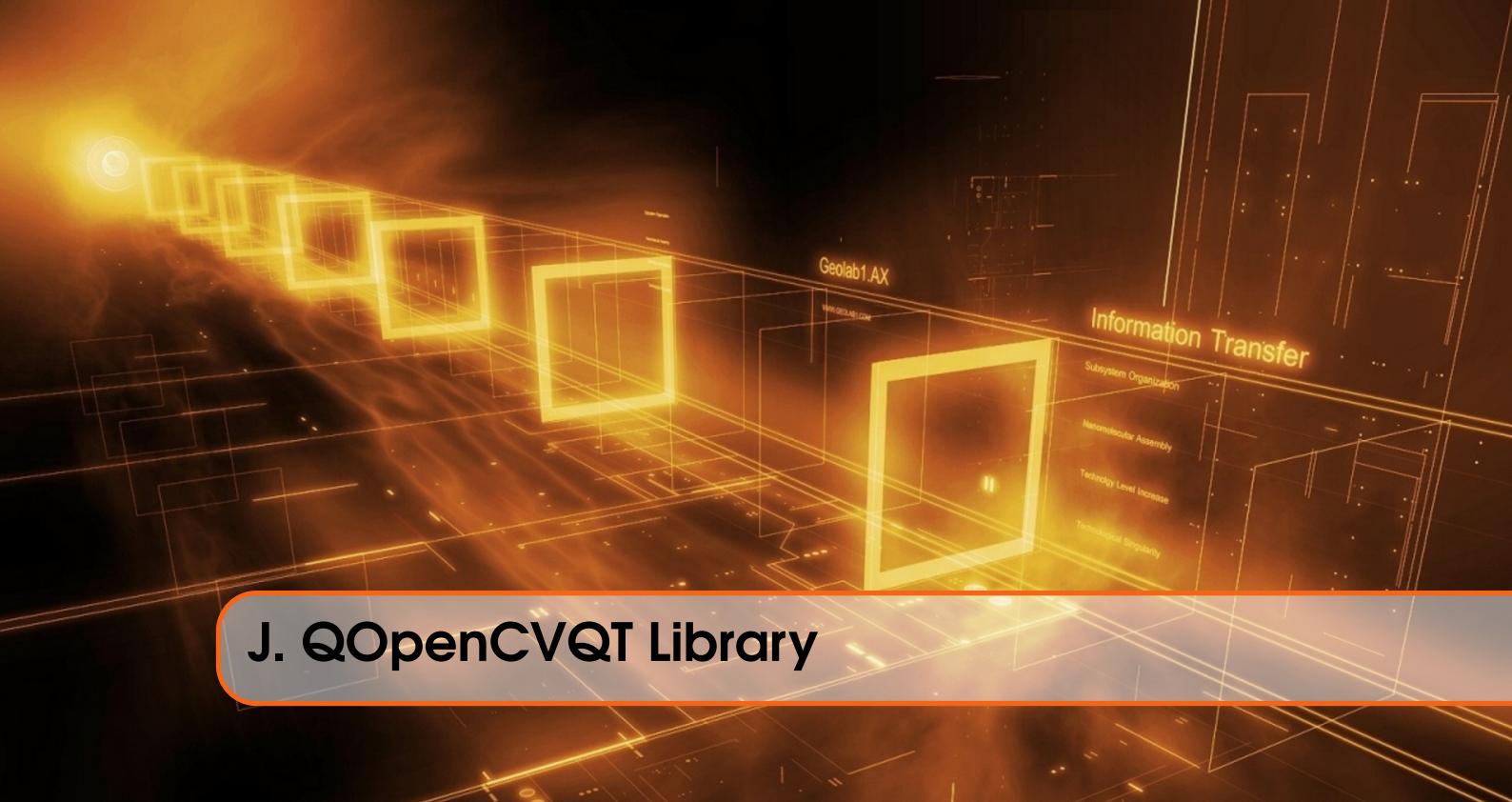
```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8 */
7 #define EXCEPTION_PARTICLE_NOT_ANALYZED "Particle not
8   analyzed Exception!"
9 #define EXCEPTION_PARTICLE_NOT_ANALYZED_N 0
9 #define EXCEPTION_NO_SNAPSHOTS "No snapshots Exception!"
10 #define EXCEPTION_NO_SNAPSHOTS_N 1

```

```
11 #define EXCEPTION_NO_IMAGES_PRESENT "No images to analyse
12 #define EXCEPTION_NO_IMAGES_PRESENT_NR 2
13
14 #pragma once
15 #include <exception>
16 #include <string>
17
18 namespace SoilAnalyzer {
19     namespace Exception {
20         class SoilAnalyzerException : public std::exception {
21             public:
22                 SoilAnalyzerException(std::string m =
23                                     EXCEPTION_PARTICLE_NOT_ANALYZED,
24                                     int n =
25                                     EXCEPTION_PARTICLE_NOT_ANALYZED_NR
26                                     ) : msg(m), nr(n) { }
27                 ~SoilAnalyzerException() _GLIBCXX_USE_NOEXCEPT {}
28                 const char *what() const _GLIBCXX_USE_NOEXCEPT {
29                     return msg.c_str(); }
30                 const int *id() const _GLIBCXX_USE_NOEXCEPT { return &
31                     nr; }
32             private:
33                 std::string msg;
34                 int nr;
35             };
36         }
37     }
38 }
```

```
1 #ifndef SOILANALYZERTYPES
2 #define SOILANALYZERTYPES
3
4 namespace SoilAnalyzer {
5     struct Point_t {
6         double x;
7         double y;
8     };
9
10    struct Lab_t {
11        float L;
12        float a;
13        float b;
14    };
15 }
16 #endif // SOILANALYZERTYPES
```



J. QOpenCVQT Library

```
1 #-----
2 #
3 # Project created by QtCreator 2015-08-08T08:11:34
4 #
5 #-----
6
7 TARGET = QOpenCVQT
8 TEMPLATE = lib
9
10 QT += gui
11
12 DEFINES += QOPENCVQT_LIBRARY
13 VERSION = 1.1.0
14 CONFIG += shared
15
16 SOURCES += qopencvqt.cpp
17
18 HEADERS += qopencvqt.h
19
20 #opencv
21 LIBS += -L/usr/local/lib -lopencv_core
22 INCLUDEPATH += /usr/local/include/opencv
23 INCLUDEPATH += /usr/local/include
24
25 #MainLib
26 unix {
27     target.path = $PWD/../../../../build/install
28     INSTALLS += target
29 }
```

```
1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
```

```

    strictly prohibited
3  * and only allowed with the written consent of the author (
   Jelle Spijker)
4  * This software is proprietary and confidential
5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6 */
7
8 #ifndef QOPENCVQT_H
9 #define QOPENCVQT_H
10
11 #include <QImage>
12 #include <opencv2/core.hpp>
13 #include <opencv2/imgproc.hpp>
14 #include <vector>
15
16 class QOpenCVQT
17 {
18 public:
19     QOpenCVQT();
20     static cv::Mat WhiteBackground(const cv::Mat &src) {
21         cv::Mat dst;
22         cv::floodFill(src, dst, cv::Point(1,1), cv::Scalar_<
23             uchar>(255,255,255));
24         return dst;
25     }
26     static QImage Mat2QImage(const cv::Mat &src) {
27         QImage dest;
28         if (src.channels() == 1) {
29             cv::Mat destRGB;
30             std::vector<cv::Mat> grayRGB(3, src);
31             cv::merge(grayRGB, destRGB);
32             dest = QImage((uchar *)destRGB.data, destRGB.cols,
33                           destRGB.rows,
34                           destRGB.step, QImage::Format_RGB888);
35         } else {
36             dest = QImage((uchar *)src.data, src.cols, src.rows,
37                           src.step,
38                           QImage::Format_RGB888);
39             dest = dest.rgbSwapped();
40         }
41     }
42 };
43 #endif // QOPENCVQT_H

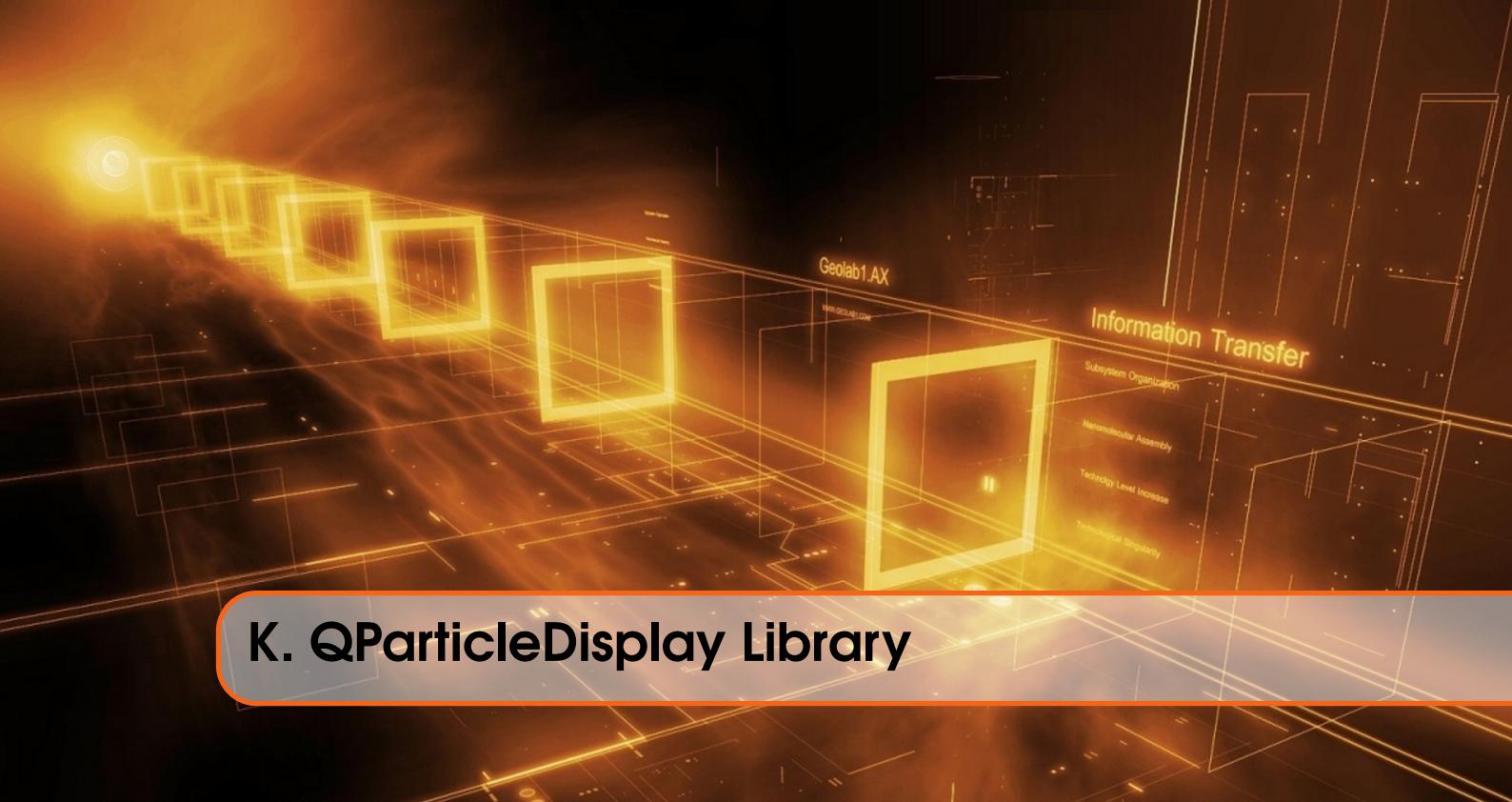
```

```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
   strictly prohibited
3  * and only allowed with the written consent of the author (
   Jelle Spijker)
4  * This software is proprietary and confidential
5  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6 */
7

```

```
8 #include "qopencvqt.h"
9
10
11 QOpenCVQT::QOpenCVQT()
12 {
13 }
```



K. QParticleDisplay Library

```
1 #-----  
2 #  
3 # Project created by QtCreator 2015-08-07T22:02:49  
4 #  
5 #-----  
6  
7 QT      += core gui concurrent  
8 QMAKE_CXXFLAGS += -std=c++11  
9  
10 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  
11  
12 TARGET = QParticleDisplay  
13 TEMPLATE = lib  
14 CONFIG += shared  
15 VERSION = 1.3.25  
16  
17 SOURCES += qparticledisplay.cpp  
18  
19 HEADERS  += qparticledisplay.h  
20  
21 FORMS    += qparticledisplay.ui  
22  
23 unix:!macx: LIBS += -L$$PWD/../../build/install/ -  
    lpictureflow-qt  
24  
25 INCLUDEPATH += $$PWD/../pictureflow-qt  
26 DEPENDPATH += $$PWD/../pictureflow-qt  
27  
28 #MainLib  
29 unix {  
30     target.path = $$PWD/../../../../build/install  
31     INSTALLS += target
```

```

32 }
33
34 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilAnalyzer
35
36 INCLUDEPATH += $$PWD/./SoilAnalyzer
37 DEPENDPATH += $$PWD/./SoilAnalyzer
38
39 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
40
41 INCLUDEPATH += $$PWD/./SoilMath
42 DEPENDPATH += $$PWD/./SoilMath
43
44 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lQOpenCVQT
45
46 INCLUDEPATH += $$PWD/./QOpenCVQT
47 DEPENDPATH += $$PWD/./QOpenCVQT
48
49 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilVision
50 INCLUDEPATH += $$PWD/./SoilVision
51 DEPENDPATH += $$PWD/./SoilVision

```

```

1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
8 */
9
10 #pragma once
11 #include <QWidget>
12 #include <QImage>
13 #include <qopencvqt.h>
14 #include <QColor>
15 #include <QWheelEvent>
16
17 namespace Ui {
18     class QParticleDisplay;
19 }
20
21 class QParticleDisplay : public QWidget
22 {
23     Q_OBJECT
24
25 public:
26     explicit QParticleDisplay(QWidget *parent = 0);
27     ~QParticleDisplay();
28     void SetSample(SoilAnalyzer::Sample *sample);
29     SoilAnalyzer::Particle *SelectedParticle;
30     void wheelEvent( QWheelEvent * event );
31     void next();

```

```

32
33 signals:
34     void particleChanged(int newValue);
35     void shapeClassificationChanged(int newValue);
36     void particleDeleted();
37
38 public slots:
39     void setSelectedParticle(int newValue);
40
41 private slots:
42     void on_selectedParticleChangedWidget(int value);
43     void on_selectedParticleChangedSlider(int value);
44     void on_pushButton_delete_clicked();
45
46 private:
47     Ui::QParticleDisplay *ui;
48     SoilAnalyzer::Sample *Sample;
49     QVector<QImage> images;
50     QImage ConvertParticleToQImage(SoilAnalyzer::Particle *
51                                     particle);
51     bool dontDoIt = false;
52 };


---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8  */
9
10
11 #include "qparticledisplay.h"
12 #include "ui_qparticledisplay.h"
13
14 QParticleDisplay::QParticleDisplay(QWidget *parent)
15     : QWidget(parent), ui(new Ui::QParticleDisplay) {
16     ui->setupUi(this);
17     ui->widget->setBackgroundColor(QColor("white"));
18     ui->widget->setSlideSize(QSize(230, 230));
19     connect(ui->widget, SIGNAL(centerIndexChanged(int)), this,
20             SLOT(on_selectedParticleChangedWidget(int)));
21     connect(ui->horizontalSlider, SIGNAL(valueChanged(int)),
22             this,
23             SLOT(on_selectedParticleChangedSlider(int)));
24 }
25
26 QParticleDisplay::~QParticleDisplay() {
27     for (uint32_t i = 0; i < ui->widget->slideCount(); i++) {
28         ui->widget->removeSlide(0);
29     }
30     delete ui->widget;
31     delete ui;
32 }
33
34 void QParticleDisplay::setSelectedParticle(int newValue) {

```

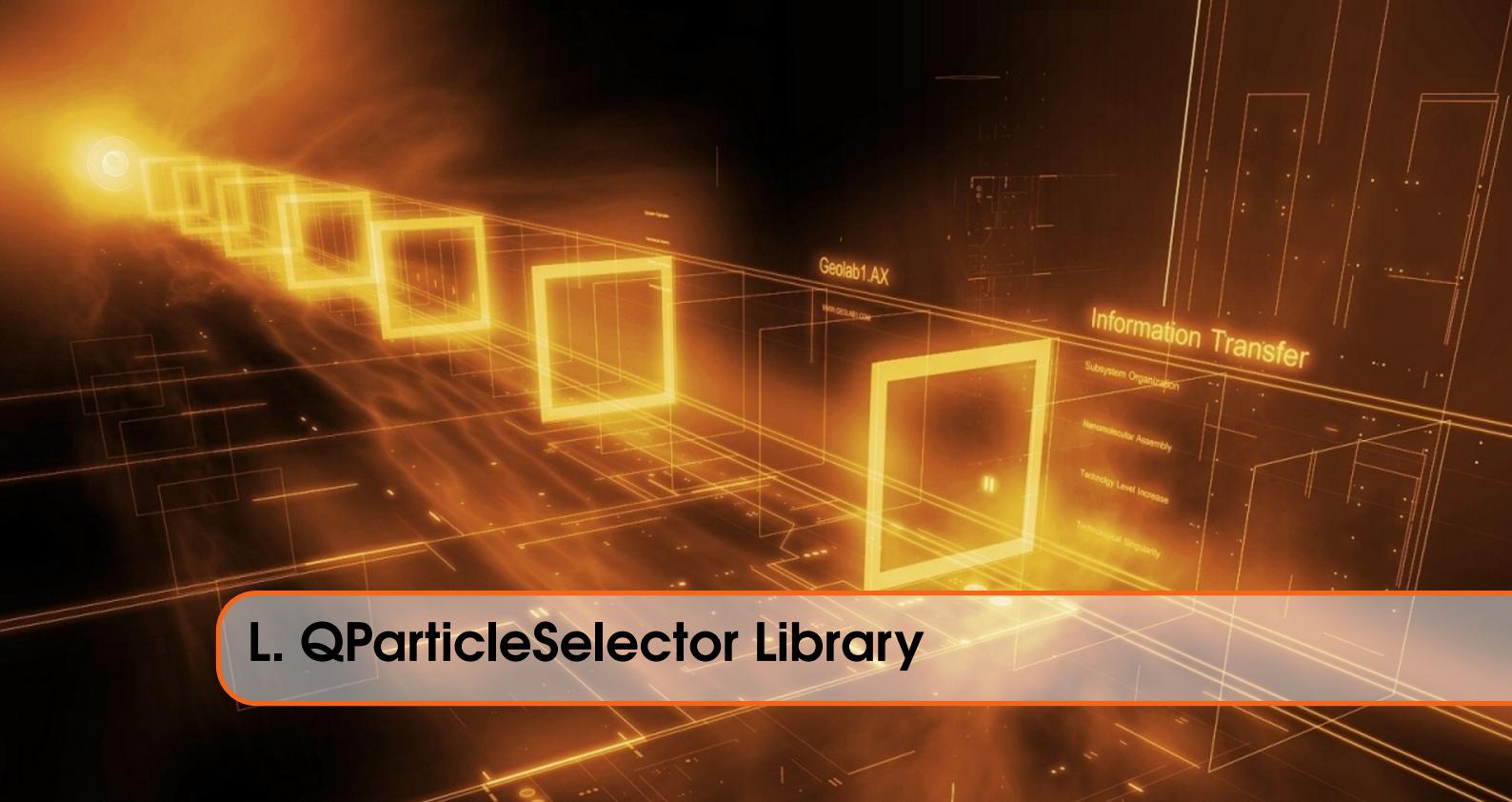
```
31     ui->widget->setCenterIndex(newValue);
32     ui->horizontalSlider->setValue(newValue);
33 }
34
35 void QParticleDisplay::SetSample(SoilAnalyzer::Sample *
36     sample) {
37     this->Sample = sample;
38     images.clear();
39     ui->widget->clear();
40     ui->horizontalSlider->setMaximum(this->Sample->
41         ParticlePopulation.size() - 1);
42     for (uint32_t i = 0; i < this->Sample->ParticlePopulation.
43         size(); i++) {
44         images.push_back(
45             ConvertParticleToQImage(&Sample->ParticlePopulation.
46                 at(i)));
47     }
48     ui->widget->addSlide(images[images.size() - 1]);
49 }
50 SelectedParticle = &Sample->ParticlePopulation[ui->widget
51     ->centerIndex()];
52 on_selectedParticleChangedSlider(0);
53 }
54
55 QImage
56 QParticleDisplay::ConvertParticleToQImage(SoilAnalyzer::
57     Particle *particle) {
58     QImage dst(particle->BW.cols + 10, particle->BW.rows + 10,
59     QImage::Format_RGB32);
60     uint32_t nData = particle->BW.cols * particle->BW.rows;
61     uint32_t sData = ((dst.width() - 1) * 5) + 5;
62     uchar *QDst = dst.bits();
63     uchar *CVBW = particle->BW.data;
64     uchar *CVRGB = particle->RGB.data;
65     for (uint32_t i = 0; i < sData; i++) {
66         *(QDst++) = 255;
67         *(QDst++) = 255;
68         *(QDst++) = 255;
69         *(QDst++) = 0;
70     }
71     for (uint32_t i = 0; i < nData; i++) {
72         if ((i % particle->BW.cols) == 0) {
73             for (uint32_t j = 0; j < 10; j++) {
74                 *(QDst++) = 255;
75                 *(QDst++) = 255;
76                 *(QDst++) = 255;
77                 *(QDst++) = 0;
78             }
79         }
80         if (CVBW[i]) {
81             *(QDst++) = *(CVRGB);
82             *(QDst++) = *(CVRGB + 1);
83             *(QDst++) = *(CVRGB + 2);
84             *(QDst++) = 0;
85             CVRGB += 3;
86         } else {
87             *(QDst++) = 255;
88         }
89     }
90 }
```

```

81         *(QDst++) = 255;
82         *(QDst++) = 255;
83         *(QDst++) = 0;
84         CVRGB += 3;
85     }
86 }
87 for (uint32_t i = 0; i < sData; i++) {
88     *(QDst++) = 255;
89     *(QDst++) = 255;
90     *(QDst++) = 255;
91     *(QDst++) = 0;
92 }
93 return dst;
94 }
95
96 void QParticleDisplay::on_pushButton_delete_clicked() {
97     Sample->ParticlePopulation.erase(Sample->
98         ParticlePopulation.begin() +
99             ui->widget->centerIndex());
100    ui->widget->removeSlide(ui->widget->centerIndex());
101    ui->horizontalSlider->setMaximum(this->Sample->
102        ParticlePopulation.size() - 1);
103    Sample->ParticleChangedStatePSD = true;
104    Sample->ParticleChangedStateAngularity = true;
105    Sample->ParticleChangedStateRoundness = true;
106    Sample->ChangesSinceLastSave = true;
107    Sample->ColorChange = true;
108    SelectedParticle = &Sample->ParticlePopulation[ui->widget
109        ->centerIndex()];
110    emit particleDeleted();
111 }
112
113 void QParticleDisplay::on_selectedParticleChangedWidget(int
114     value) {
115     if (!dontDoIt) {
116         dontDoIt = true;
117         ui->horizontalSlider->setValue(value);
118         SelectedParticle = &Sample->ParticlePopulation[ui->
119             widget->centerIndex()];
120         QString volume;
121         volume.sprintf("%+06.2f", SelectedParticle->
122             GetSiDiameter());
123         ui->label_Volume->setText(volume);
124         emit particleChanged(value);
125         emit shapeClassificationChanged(SelectedParticle->
126             Classification.Category);
127         dontDoIt = false;
128     }
129 }
130
131 void QParticleDisplay::on_selectedParticleChangedSlider(int
132     value) {
133     if (!dontDoIt) {
134         dontDoIt = true;
135         ui->widget->setCenterIndex(value);
136     }
137 }

```

```
128     SelectedParticle = &Sample->ParticlePopulation[ui->
129         widget->centerIndex()];
130     QString volume;
131     volume.sprintf("%+06.2f", SelectedParticle->
132         GetSiDiameter());
133     ui->label_Volume->setText(volume);
134     emit particleChanged(value);
135     emit shapeClassificationChanged(SelectedParticle->
136         Classification.Category);
137     dontDoIt = false;
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 void QParticleDisplay::wheelEvent(QWheelEvent *event) {
151     int i = ui->widget->centerIndex();
152     i -= event->delta() / 120;
153     if (i < 0) {
154         i = ui->widget->slideCount() - abs(i) - 1;
155     } else if (i >= ui->widget->slideCount()) {
156         i = 0;
157     }
158     ui->widget->setCenterIndex(i);
159     on_selectedParticleChangedWidget(i);
160 }
```



L. QParticleSelector Library

```
1 #-----  
2 #  
3 # Project created by QtCreator 2015-08-07T18:56:27  
4 #  
5 #-----  
6  
7 QT      += core gui  
8 QMAKE_CXXFLAGS += -std=c++11  
9  
10 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  
11  
12 TARGET = QParticleSelector  
13 TEMPLATE = lib  
14 CONFIG += shared  
15 VERSION = 0.1.11  
16  
17 SOURCES += qparticleselector.cpp  
18  
19 HEADERS  += qparticleselector.h  
20  
21 FORMS    += qparticleselector.ui  
22  
23 RESOURCES += \  
24     qparticleselector.qrc  
25  
26 #MainLib  
27 unix {  
28     target.path = $PWD/../../../../build/install  
29     INSTALLS += target  
30 }  
31  
32 unix:!macx: LIBS += -L$$PWD/../../../../build/install/ -lSoilMath
```

```
33
34 INCLUDEPATH += $$PWD/../SoilMath
35 DEPENDPATH += $$PWD/../SoilMath


---


1 #ifndef QPARTICLESELECTOR_H
2 #define QPARTICLESELECTOR_H
3
4 #include <QWidget>
5 #include <QPushButton>
6
7 namespace Ui {
8     class QParticleSelector;
9 }
10
11 class QParticleSelector : public QWidget
12 {
13     Q_OBJECT
14
15 public:
16     explicit QParticleSelector(QWidget *parent = 0);
17     ~QParticleSelector();
18
19     void setDisabled(bool value, int currentClass = 1);
20
21 signals:
22     void valueChanged(int newValue);
23
24 public slots:
25     void setValue(int newValue);
26
27 private slots:
28     void on_pb_1_clicked(bool checked);
29
30     void on_pb_2_clicked(bool checked);
31
32     void on_pb_3_clicked(bool checked);
33
34     void on_pb_4_clicked(bool checked);
35
36     void on_pb_5_clicked(bool checked);
37
38     void on_pb_6_clicked(bool checked);
39
40     void on_pb_7_clicked(bool checked);
41
42     void on_pb_8_clicked(bool checked);
43
44     void on_pb_9_clicked(bool checked);
45
46     void on_pb_10_clicked(bool checked);
47
48     void on_pb_11_clicked(bool checked);
49
50     void on_pb_12_clicked(bool checked);
51
52     void on_pb_13_clicked(bool checked);
```

```
53
54     void on_pb_14_clicked(bool checked);
55
56     void on_pb_15_clicked(bool checked);
57
58     void on_pb_16_clicked(bool checked);
59
60     void on_pb_17_clicked(bool checked);
61
62     void on_pb_18_clicked(bool checked);
63
64 private:
65     QVector<QPushButton *> btns;
66     Ui::QParticleSelector *ui;
67 };
68
69 #endif // QPARTICLESELECTOR_H


---

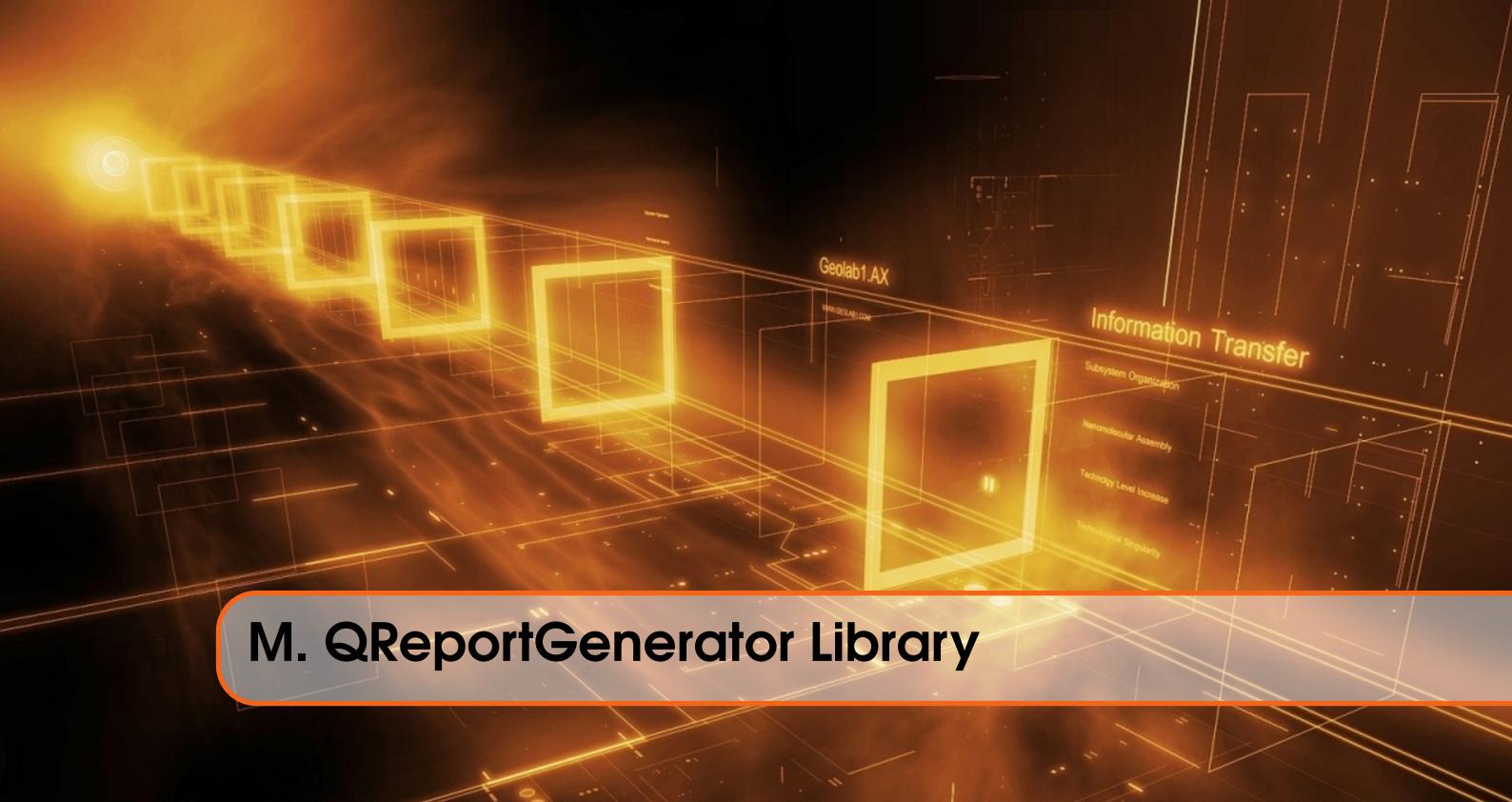

1 #include "qparticleselector.h"
2 #include "ui_qparticleselector.h"
3
4 QParticleSelector::QParticleSelector(QWidget *parent)
5     : QWidget(parent), ui(new Ui::QParticleSelector) {
6     ui->setupUi(this);
7     btns.push_back(ui->pb_1);
8     btns.push_back(ui->pb_2);
9     btns.push_back(ui->pb_3);
10    btns.push_back(ui->pb_4);
11    btns.push_back(ui->pb_5);
12    btns.push_back(ui->pb_6);
13    btns.push_back(ui->pb_7);
14    btns.push_back(ui->pb_8);
15    btns.push_back(ui->pb_9);
16    btns.push_back(ui->pb_10);
17    btns.push_back(ui->pb_11);
18    btns.push_back(ui->pb_12);
19    btns.push_back(ui->pb_13);
20    btns.push_back(ui->pb_14);
21    btns.push_back(ui->pb_15);
22    btns.push_back(ui->pb_16);
23    btns.push_back(ui->pb_17);
24    btns.push_back(ui->pb_18);
25 }
26
27 QParticleSelector::~QParticleSelector() {
28     for (auto b : btns) {
29         delete b;
30     }
31     btns.clear();
32     delete ui;
33 }
34
35 void QParticleSelector::setValue(int newValue) {
36     btns[newValue - 1]->setChecked(true);
37 }
```

```
39 void QParticleSelector::setDisabled(bool value, int
40     currentClass) {
41     for (auto b : btns) {
42         b->setDisabled(value);
43     }
44     if (currentClass > 18 || currentClass < 1) {
45         bns[0]->setChecked(true);
46     } else {
47         bns[currentClass - 1]->setChecked(true);
48     }
49 }
50 void QParticleSelector::on_pb_1_clicked(bool checked) {
51     if (checked) {
52         emit valueChanged(1);
53     }
54 }
55
56 void QParticleSelector::on_pb_2_clicked(bool checked) {
57     if (checked) {
58         emit valueChanged(2);
59     }
60 }
61
62 void QParticleSelector::on_pb_3_clicked(bool checked) {
63     if (checked) {
64         emit valueChanged(3);
65     }
66 }
67
68 void QParticleSelector::on_pb_4_clicked(bool checked) {
69     if (checked) {
70         emit valueChanged(4);
71     }
72 }
73
74 void QParticleSelector::on_pb_5_clicked(bool checked) {
75     if (checked) {
76         emit valueChanged(5);
77     }
78 }
79
80 void QParticleSelector::on_pb_6_clicked(bool checked) {
81     if (checked) {
82         emit valueChanged(6);
83     }
84 }
85
86 void QParticleSelector::on_pb_7_clicked(bool checked) {
87     if (checked) {
88         emit valueChanged(7);
89     }
90 }
91
92 void QParticleSelector::on_pb_8_clicked(bool checked) {
93     if (checked) {
```

```
94     emit valueChanged(8);
95 }
96 }
97
98 void QParticleSelector::on_pb_9_clicked(bool checked) {
99     if (checked) {
100         emit valueChanged(9);
101     }
102 }
103
104 void QParticleSelector::on_pb_10_clicked(bool checked) {
105     if (checked) {
106         emit valueChanged(10);
107     }
108 }
109
110 void QParticleSelector::on_pb_11_clicked(bool checked) {
111     if (checked) {
112         emit valueChanged(11);
113     }
114 }
115
116 void QParticleSelector::on_pb_12_clicked(bool checked) {
117     if (checked) {
118         emit valueChanged(12);
119     }
120 }
121
122 void QParticleSelector::on_pb_13_clicked(bool checked) {
123     if (checked) {
124         emit valueChanged(13);
125     }
126 }
127
128 void QParticleSelector::on_pb_14_clicked(bool checked) {
129     if (checked) {
130         emit valueChanged(14);
131     }
132 }
133
134 void QParticleSelector::on_pb_15_clicked(bool checked) {
135     if (checked) {
136         emit valueChanged(15);
137     }
138 }
139
140 void QParticleSelector::on_pb_16_clicked(bool checked) {
141     if (checked) {
142         emit valueChanged(16);
143     }
144 }
145
146 void QParticleSelector::on_pb_17_clicked(bool checked) {
147     if (checked) {
148         emit valueChanged(17);
149     }

```

```
150  }
151
152 void QParticleSelector::on_pb_18_clicked(bool checked) {
153     if (checked) {
154         emit valueChanged(18);
155     }
156 }
```



M. QReportGenerator Library

```
1 #-----  
2 #  
3 # Project created by QtCreator 2015-08-20T08:46:42  
4 #  
5 #-----  
6  
7 QT      += core gui concurrent network  
8 QMAKE_CXXFLAGS += -std=c++11  
9  
10 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport  
     multimedia multimediawidgets  
11  
12 @  
13 CONFIG(release, debug|release):DEFINES += QT_NO_DEBUG_OUTPUT  
14 @  
15  
16 unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../../../build/install/  
17  
18 TARGET = QReportGenerator  
19 TEMPLATE = lib  
20 CONFIG += shared  
21 VERSION = 0.1.00  
22  
23 SOURCES += \  
24     qreportgenerator.cpp \  
25     ../../qcustomplot/examples/text-document-integration/  
         qcpdocumentobject.cpp  
26  
27 HEADERS  += \  
28     qreportgenerator.h \  
29     ../../qcustomplot/examples/text-document-integration/  
         qcpdocumentobject.h
```

```

30
31 FORMS     += \
32     qreportgenerator.ui
33
34 #MainLib
35 unix {
36     target.path = $PWD/../../build/install
37     INSTALLS += target
38 }
39
40 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
41 INCLUDEPATH += $$PWD/../SoilMath
42 DEPENDPATH += $$PWD/../SoilMath
43
44 DEFINES += QCUSTOMPLOT_USE_LIBRARY
45 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
    lqcustomplot
46
47 INCLUDEPATH += $$PWD/../qcustomplot
48 DEPENDPATH += $$PWD/../qcustomplot
49
50 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
    lSoilAnalyzer
51 INCLUDEPATH += $$PWD/../SoilAnalyzer
52 DEPENDPATH += $$PWD/../SoilAnalyzer
53
54 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
    lSoilVision
55 INCLUDEPATH += $$PWD/../SoilVision
56 DEPENDPATH += $$PWD/../SoilVision
57
58 RESOURCES += \
59     qreportresources.qrc \
60     ../VSA/vsa_resources.qrc
61
62 #maps
63 Mapstarget.path += $$OUT_PWD/Maps
64 Mapstarget.files += $$PWD/Maps/*
65 INSTALLS += Mapstarget
66 bMapstarget.path += $$PWD/../../build/install/Maps
67 bMapstarget.files += $$PWD/Maps/*
68 INSTALLS += bMapstarget

```

```

1 #ifndef QREPORTGENERATOR_H
2 #define QREPORTGENERATOR_H
3
4 #include <QMainWindow>
5 #include <QTextDocument>
6 #include <QDebug>
7 #include <QTextBlockFormat>
8 #include <QTextCharFormat>
9 #include <QTextBlock>
10 #include <QNetworkAccessManager>
11 #include <QNetworkReply>
12 #include <QTextDocumentWriter>
13 #include <QPrinter>

```

```
14
15 #include "soilanalyzer.h"
16 #include "SoilMath.h"
17
18 #include <qcustomplot.h>
19 #include "../qcustomplot/examples/text-document-integration/
20     qcpdocumentobject.h"
21
22 namespace Ui {
23     class QReportGenerator;
24 }
25
26 class QReportGenerator : public QMainWindow
27 {
28     Q_OBJECT
29
30     public:
31         QTextDocument *Report = nullptr;
32         SoilAnalyzer::Sample *Sample = nullptr;
33         SoilAnalyzer::SoilSettings *Settings = nullptr;
34         QCustomePlot *PSD = nullptr;
35         QCustomePlot *Roundness = nullptr;
36         QCustomePlot *Angularity = nullptr;
37
38     explicit QReportGenerator(QWidget *parent = 0,
39         SoilAnalyzer::Sample *sample = nullptr, SoilAnalyzer::
40         SoilSettings *settings = nullptr, QCustomePlot *psd =
41         nullptr, QCustomePlot *roundness = nullptr, QCustomePlot
42         *angularity = nullptr);
43     ~QReportGenerator();
44
45     private slots:
46         void on_locationImageDownloaded(QNetworkReply *reply);
47
48         void on_actionSave_triggered();
49
50         void on_actionExport_to_PDF_triggered();
51
52     private:
53         Ui::QReportGenerator *ui;
54         QCustomePlot *CIELabPlot = nullptr;
55
56         QImage *mapLocation = nullptr;
57
58         QTextCursor rCurs;
59
60         // Layout formats
61         QTextBlockFormat TitleFormat;
62         QTextBlockFormat HeaderFormat;
63         QTextBlockFormat GeneralFormat;
64         QTextBlockFormat ImageGraphFormat;
65
66         QTextCharFormat TitleTextFormat;
```

```
65     QTextCharFormat HeaderTextFormat;
66     QTextCharFormat GtxtFormat;
67     QTextCharFormat GFieldtxtFormat;
68
69     QTextListFormat GeneralSampleList;
70     QTextTableFormat GeneralTextTableFormat;
71
72
73     QFont TitleFont;
74     QFont HeaderFont;
75     QFont GeneralFont;
76     QFont FieldFont;
77 }
78
79 #endif // QREPORTGENERATOR_H
```

```
1 #include "qreportgenerator.h"
2 #include "ui_qreportgenerator.h"
3
4 QReportGenerator::QReportGenerator(QWidget *parent,
5                                     SoilAnalyzer::Sample *
6                                     sample,
7                                     SoilAnalyzer::
8                                     SoilSettings *settings
9                                     ,
10                                     QCustomPlot *psd,
11                                     QCustomPlot *roundness
12                                     ,
13                                     QCustomPlot *angularity)
14     : QMainWindow(parent), ui(new Ui::QReportGenerator) {
15     ui->setupUi(this);
16     if (settings == nullptr) {
17         settings = new SoilAnalyzer::SoilSettings;
18     }
19     this->Settings = settings;
20     if (sample == nullptr) {
21         sample = new SoilAnalyzer::Sample;
22     }
23     this->Sample = sample;
24
25     if (psd == nullptr) {
26         psd = new QCustomPlot;
27     }
28     this->PSD = psd;
29
30     if (roundness == nullptr) {
31         roundness = new QCustomPlot;
32     }
33     this->Roundness = roundness;
34
35     if (angularity == nullptr) {
36         angularity = new QCustomPlot;
37     }
38     this->Angularity = angularity;
39
40     Report = new QTextDocument(ui->textEdit);
```

```
36     ui->textEdit->setDocument(Report);
37     rCurs = QTextCursor(Report);
38
39     // Setup the layout
40     TitleFormat.setAlignment(Qt::AlignCenter);
41     TitleFont.setBold(true);
42     TitleFont.setPointSize(36);
43     TitleTextFormat.setFont(TitleFont);
44
45     HeaderFormat.setAlignment(Qt::AlignCenter);
46     HeaderFormat.setPageBreakPolicy(QTextFormat::
47         PageBreak_AlwaysBefore);
47     HeaderFormat.setTopMargin(40);
48     HeaderFormat.setBottomMargin(10);
49     HeaderFont.setBold(true);
50     HeaderFont.setPointSize(18);
51     HeaderTextFormat.setFont(HeaderFont);
52
53     ImageGraphFormat.setAlignment(Qt::AlignCenter);
54     ImageGraphFormat.setTopMargin(10);
55     ImageGraphFormat.setBottomMargin(10);
56
57     GeneralFormat.setAlignment(Qt::AlignLeft);
58
59     GeneralFont.setPointSize(12);
60     GeneralFont.setBold(false);
61     GtxtFormat.setFont(GeneralFont);
62
63     FieldFont.setBold(true);
64     GFieldtxtFormat.setFont(FieldFont);
65
66     GeneralSampleList.setStyle(QTextListFormat::ListDisc);
67
68     GeneralTextTableFormat.setHeaderRowCount(1);
69     GeneralTextTableFormat.setBorderStyle(QTextFrameFormat::
70         BorderStyle_None);
70     GeneralTextTableFormat.setWidth(
71         QTextLength(QTextLength::PercentageLength, 90));
72     GeneralTextTableFormat.setAlignment(Qt::AlignCenter);
73
74     // Setup the Title
75     rCurs.setBlockFormat(TitleFormat);
76     rCurs.insertText("Soil Report", TitleTextFormat);
77     rCurs.insertBlock();
78
79     // Setup the general Text
80     rCurs.insertBlock(ImageGraphFormat);
81     QTextTable *mainTable = rCurs.insertTable(5, 2,
82         GeneralTextTableFormat);
82     rCurs = mainTable->cellAt(0, 0).firstCursorPosition();
83     rCurs.insertText("Sample name:", GFieldtxtFormat);
84     rCurs.movePosition(QTextCursor::NextCell);
85     rCurs.insertText(QString::fromStdString(Sample->Name),
86         GtxtFormat);
86     rCurs.movePosition(QTextCursor::NextCell);
87
```

```

88 rCurs.insertText("Sample ID:", GFieldtxtFormat);
89 rCurs.movePosition(QTextCursor::NextCell);
90 rCurs.insertText(QString::number(Sample->ID), GtxtFormat);
91 rCurs.movePosition(QTextCursor::NextCell);
92
93 rCurs.insertText("Date:", GFieldtxtFormat);
94 rCurs.movePosition(QTextCursor::NextCell);
95 rCurs.insertText(QString::fromStdString(Sample->Date),
96                 GtxtFormat);
97 rCurs.movePosition(QTextCursor::NextCell);
98
99 rCurs.insertText("Location:", GFieldtxtFormat);
100 rCurs.movePosition(QTextCursor::NextCell);
101 rCurs.insertText(QString::number(Sample->Latitude),
102                  GtxtFormat);
103 rCurs.movePosition(QTextCursor::NextCell);
104
105 rCurs.insertText("Sample depth:", GFieldtxtFormat);
106 rCurs.movePosition(QTextCursor::NextCell);
107 rCurs.insertText(QString::number(Sample->Depth),
108                  GtxtFormat);
109 rCurs.insertText(" [m]", GtxtFormat);
110 rCurs.movePosition(QTextCursor::NextBlock);
111 rCurs.insertBlock();
112
113 // Insert the Google map
114 getLocationMap(Sample->Latitude, Sample->Longitude);
115
116 // Setup the QCUSTOMplot handler
117 QCPDocumentObject *plotObjectHandler = new
118     QCPDocumentObject(this);
119 ui->textEdit->document()->documentLayout()->
120     registerHandler(
121         QCPDocumentObject::PlotTextFormat, plotObjectHandler);
122
123 // Setup the Textdata for the PSD
124 rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
125 rCurs.insertText("Particle Size Distribution");
126
127 rCurs.insertBlock(ImageGraphFormat);
128 QTextTable *PSDdescr = rCurs.insertTable(6, 2,
129                                         GeneralTextTableFormat);
130 rCurs = PSDdescr->cellAt(0, 0).firstCursorPosition();
131 rCurs.insertText("No of particles:", GFieldtxtFormat);
132 rCurs.movePosition(QTextCursor::NextCell);
133 rCurs.insertText(QString::number(Sample->PSD.n),
134                 GtxtFormat);
135 rCurs.movePosition(QTextCursor::NextCell);
136
137 rCurs.insertText("Mean: ", GFieldtxtFormat);
138 rCurs.movePosition(QTextCursor::NextCell);
139 rCurs.insertText(QString::number(Sample->PSD.Mean),
140                 GtxtFormat);

```

```

135     rCurs.movePosition(QTextCursor::NextCell);
136
137     rCurs.insertText("Minimum: ", GFieldtxtFormat);
138     rCurs.movePosition(QTextCursor::NextCell);
139     rCurs.insertText(QString::number(Sample->PSD.min),
140                     GtxtFormat);
140     rCurs.movePosition(QTextCursor::NextCell);
141
142     rCurs.insertText("Maximum: ", GFieldtxtFormat);
143     rCurs.movePosition(QTextCursor::NextCell);
144     rCurs.insertText(QString::number(Sample->PSD.max),
145                     GtxtFormat);
145     rCurs.movePosition(QTextCursor::NextCell);
146
147     rCurs.insertText("Range: ", GFieldtxtFormat);
148     rCurs.movePosition(QTextCursor::NextCell);
149     rCurs.insertText(QString::number(Sample->PSD.Range),
150                     GtxtFormat);
150     rCurs.movePosition(QTextCursor::NextCell);
151
152     rCurs.insertText("Standard deviation: ", GFieldtxtFormat);
153     rCurs.movePosition(QTextCursor::NextCell);
154     rCurs.insertText(QString::number(Sample->PSD.Std),
155                     GtxtFormat);
155     rCurs.movePosition(QTextCursor::NextBlock);
156
157 // Setup the PSD
158 rCurs.insertBlock(ImageGraphFormat);
159 rCurs.insertText(QString(QChar::ObjectReplacementCharacter
160                     ),
161                     QCPDocumentObject::generatePlotFormat(PSD
162                     , 600, 350));
163
164 rCurs.insertBlock(ImageGraphFormat);
165 QTextTable *PSDdata = rCurs.insertTable(16, 3,
166                                         GeneralTextTableFormat);
167 rCurs.insertText("Mesh Size [mm]", GFieldtxtFormat);
168 rCurs.movePosition(QTextCursor::NextCell);
169 rCurs.insertText("Cummulatief [%]", GFieldtxtFormat);
170 rCurs.movePosition(QTextCursor::NextCell);
171 rCurs.insertText("Retained [-]", GFieldtxtFormat);
172 rCurs.movePosition(QTextCursor::NextCell);
173 rCurs.insertText("2", GFieldtxtFormat);
174 rCurs.movePosition(QTextCursor::NextCell);
175 rCurs.insertText(QString::number(Sample->PSD.CFD[14]),
176                     GtxtFormat);
177 rCurs.movePosition(QTextCursor::NextCell);
178 rCurs.insertText("1.4", GFieldtxtFormat);
179 rCurs.movePosition(QTextCursor::NextCell);
180 rCurs.insertText(QString::number(Sample->PSD.CFD[13]),
181                     GtxtFormat);
182 rCurs.movePosition(QTextCursor::NextCell);
183 rCurs.insertText(QString::number(Sample->PSD.bins[14]),
184                     GtxtFormat);
185 rCurs.movePosition(QTextCursor::NextCell);
186 rCurs.insertText("1.4", GFieldtxtFormat);
187 rCurs.movePosition(QTextCursor::NextCell);
188 rCurs.insertText(QString::number(Sample->PSD.CFD[13]),
189                     GtxtFormat);
190 rCurs.movePosition(QTextCursor::NextCell);
191 rCurs.insertText(QString::number(Sample->PSD.bins[13]),
192                     GtxtFormat);

```

```
        GtxtFormat);
181 rCurs.movePosition(QTextCursor::NextCell);
182 rCurs.insertText("1", GFieldtxtFormat);
183 rCurs.movePosition(QTextCursor::NextCell);
184 rCurs.insertText(QString::number(Sample->PSD.CFD[12]),
185     GtxtFormat);
186 rCurs.movePosition(QTextCursor::NextCell);
187 rCurs.insertText(QString::number(Sample->PSD.bins[12]),
188     GtxtFormat);
189 rCurs.movePosition(QTextCursor::NextCell);
190 rCurs.insertText(QString::number(Sample->PSD.CFD[11]),
191     GtxtFormat);
192 rCurs.movePosition(QTextCursor::NextCell);
193 rCurs.insertText(QString::number(Sample->PSD.bins[11]),
194     GtxtFormat);
195 rCurs.movePosition(QTextCursor::NextCell);
196 rCurs.insertText(QString::number(Sample->PSD.CFD[10]),
197     GtxtFormat);
198 rCurs.movePosition(QTextCursor::NextCell);
199 rCurs.insertText(QString::number(Sample->PSD.bins[10]),
200     GtxtFormat);
201 rCurs.movePosition(QTextCursor::NextCell);
202 rCurs.insertText(QString::number(Sample->PSD.CFD[9]),
203     GtxtFormat);
204 rCurs.movePosition(QTextCursor::NextCell);
205 rCurs.insertText(QString::number(Sample->PSD.bins[9]),
206     GtxtFormat);
207 rCurs.movePosition(QTextCursor::NextCell);
208 rCurs.insertText(QString::number(Sample->PSD.CFD[8]),
209     GtxtFormat);
210 rCurs.movePosition(QTextCursor::NextCell);
211 rCurs.insertText(QString::number(Sample->PSD.bins[8]),
212     GtxtFormat);
213 rCurs.movePosition(QTextCursor::NextCell);
214 rCurs.insertText(QString::number(Sample->PSD.CFD[7]),
215     GtxtFormat);
216 rCurs.movePosition(QTextCursor::NextCell);
217 rCurs.insertText(QString::number(Sample->PSD.bins[7]),
218     GtxtFormat);
219 rCurs.movePosition(QTextCursor::NextCell);
220 rCurs.insertText(QString::number(Sample->PSD.CFD[6]),
221     GtxtFormat);
222 rCurs.movePosition(QTextCursor::NextCell);
223 rCurs.insertText(QString::number(Sample->PSD.bins[6]),
```

```

        GtxtFormat);
223 rCurs.movePosition(QTextCursor::NextCell);
224 rCurs.insertText("0.09", GFieldtxtFormat);
225 rCurs.movePosition(QTextCursor::NextCell);
226 rCurs.insertText(QString::number(Sample->PSD.CFD[5]),
        GtxtFormat);
227 rCurs.movePosition(QTextCursor::NextCell);
228 rCurs.insertText(QString::number(Sample->PSD.bins[5]),
        GtxtFormat);
229 rCurs.movePosition(QTextCursor::NextCell);
230 rCurs.insertText("0.075", GFieldtxtFormat);
231 rCurs.movePosition(QTextCursor::NextCell);
232 rCurs.insertText(QString::number(Sample->PSD.CFD[4]),
        GtxtFormat);
233 rCurs.movePosition(QTextCursor::NextCell);
234 rCurs.insertText(QString::number(Sample->PSD.bins[4]),
        GtxtFormat);
235 rCurs.movePosition(QTextCursor::NextCell);
236 rCurs.insertText("0.063", GFieldtxtFormat);
237 rCurs.movePosition(QTextCursor::NextCell);
238 rCurs.insertText(QString::number(Sample->PSD.CFD[3]),
        GtxtFormat);
239 rCurs.movePosition(QTextCursor::NextCell);
240 rCurs.insertText(QString::number(Sample->PSD.bins[3]),
        GtxtFormat);
241 rCurs.movePosition(QTextCursor::NextCell);
242 rCurs.insertText("0.045", GFieldtxtFormat);
243 rCurs.movePosition(QTextCursor::NextCell);
244 rCurs.insertText(QString::number(Sample->PSD.CFD[2]),
        GtxtFormat);
245 rCurs.movePosition(QTextCursor::NextCell);
246 rCurs.insertText(QString::number(Sample->PSD.bins[2]),
        GtxtFormat);
247 rCurs.movePosition(QTextCursor::NextCell);
248 rCurs.insertText("0.038", GFieldtxtFormat);
249 rCurs.movePosition(QTextCursor::NextCell);
250 rCurs.insertText(QString::number(Sample->PSD.CFD[1]),
        GtxtFormat);
251 rCurs.movePosition(QTextCursor::NextCell);
252 rCurs.insertText(QString::number(Sample->PSD.bins[1]),
        GtxtFormat);
253 rCurs.movePosition(QTextCursor::NextCell);
254 rCurs.insertText("0", GFieldtxtFormat);
255 rCurs.movePosition(QTextCursor::NextCell);
256 rCurs.insertText(QString::number(Sample->PSD.CFD[0]),
        GtxtFormat);
257 rCurs.movePosition(QTextCursor::NextCell);
258 rCurs.insertText(QString::number(Sample->PSD.bins[0]),
        GtxtFormat);
259 rCurs.movePosition(QTextCursor::NextBlock);
260
261 // Setup the Textdata for the Roundness
262 rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
263 rCurs.insertText("Sphericity Classification");
264
265 rCurs.insertBlock(ImageGraphFormat);

```

```

266     QTextTable *Rounddescr = rCurs.insertTable(6, 2,
267         GeneralTextTableFormat);
268     rCurs = Rounddescr->cellAt(0, 0).firstCursorPosition();
269     rCurs.insertText("No of particles:", GFieldtxtFormat);
270     rCurs.movePosition(QTextCursor::NextCell);
271     rCurs.insertText(QString::number(Sample->Roundness.n),
272         GtxtFormat);
273     rCurs.movePosition(QTextCursor::NextCell);
274
275     rCurs.insertText("Mean: ", GFieldtxtFormat);
276     rCurs.movePosition(QTextCursor::NextCell);
277     rCurs.insertText(QString::number(Sample->Roundness.Mean),
278         GtxtFormat);
279     rCurs.movePosition(QTextCursor::NextCell);
280
281     rCurs.insertText("Minimum: ", GFieldtxtFormat);
282     rCurs.movePosition(QTextCursor::NextCell);
283     rCurs.insertText(QString::number(Sample->Roundness.min),
284         GtxtFormat);
285     rCurs.movePosition(QTextCursor::NextCell);
286
287     rCurs.insertText("Maximum: ", GFieldtxtFormat);
288     rCurs.movePosition(QTextCursor::NextCell);
289     rCurs.insertText(QString::number(Sample->Roundness.max),
290         GtxtFormat);
291     rCurs.movePosition(QTextCursor::NextCell);
292
293     rCurs.insertText("Standard deviation: ", GFieldtxtFormat);
294     rCurs.movePosition(QTextCursor::NextCell);
295     rCurs.insertText(QString::number(Sample->Roundness.Std),
296         GtxtFormat);
297     rCurs.movePosition(QTextCursor::NextBlock);
298
299     // Setup the Roundness Graph
300     rCurs.insertBlock(ImageGraphFormat);
301     rCurs.insertText(QString(QChar::ObjectReplacementCharacter
302         ),
303         QCPDocumentObject::generatePlotFormat(
304             Roundness, 600, 400));
305
306     // Setup the Textdata for the Roundness
307     rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
308     rCurs.insertText("Angularity Classification");
309
310     rCurs.insertBlock(ImageGraphFormat);
311     QTextTable *Angularitydescr = rCurs.insertTable(6, 2,
312         GeneralTextTableFormat);
313     rCurs = Angularitydescr->cellAt(0, 0).firstCursorPosition
314         ();
315     rCurs.insertText("No of particles:", GFieldtxtFormat);

```

```

311     rCurs.movePosition(QTextCursor::NextCell);
312     rCurs.insertText(QString::number(Sample->Angularity.n),
313                     GtxtFormat);
314     rCurs.movePosition(QTextCursor::NextCell);
315     rCurs.insertText("Mean: ", GFieldtxtFormat);
316     rCurs.movePosition(QTextCursor::NextCell);
317     rCurs.insertText(QString::number(Sample->Angularity.Mean),
318                     GtxtFormat);
319     rCurs.movePosition(QTextCursor::NextCell);
320     rCurs.insertText("Minimum: ", GFieldtxtFormat);
321     rCurs.movePosition(QTextCursor::NextCell);
322     rCurs.insertText(QString::number(Sample->Angularity.min),
323                     GtxtFormat);
324     rCurs.movePosition(QTextCursor::NextCell);
325     rCurs.insertText("Maximum: ", GFieldtxtFormat);
326     rCurs.movePosition(QTextCursor::NextCell);
327     rCurs.insertText(QString::number(Sample->Angularity.max),
328                     GtxtFormat);
328     rCurs.movePosition(QTextCursor::NextCell);
329
330     rCurs.insertText("Range: ", GFieldtxtFormat);
331     rCurs.movePosition(QTextCursor::NextCell);
332     rCurs.insertText(QString::number(Sample->Angularity.Range)
333                     , GtxtFormat);
333     rCurs.movePosition(QTextCursor::NextCell);
334
335     rCurs.insertText("Standard deviation: ", GFieldtxtFormat);
336     rCurs.movePosition(QTextCursor::NextCell);
337     rCurs.insertText(QString::number(Sample->Angularity.Std),
338                     GtxtFormat);
338     rCurs.movePosition(QTextCursor::NextBlock);
339
340 // Setup the Roundness Graph
341 rCurs.insertBlock(ImageGraphFormat);
342 rCurs.insertText(QString(QChar::ObjectReplacementCharacter
343                     ),
344                     QCPDocumentObject::generatePlotFormat(
345                     Angularity, 600, 400));
346
347 // Setup the CIE La*b* graph
348 // Setup the Textdata for the Roundness
349 rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
350 rCurs.insertText("CIE La*b*");
351
352 SetupCIELabPlot();
353 rCurs.insertBlock(ImageGraphFormat);
354 rCurs.insertText(QString(QChar::ObjectReplacementCharacter
355                     ),
356                     QCPDocumentObject::generatePlotFormat(
357                     CIELabPlot, 600, 400));
358
359 }
360

```

```

357 void QReportGenerator::getLocationMap(double &latitude,
358                                         double &longitude) {
359     QNetworkAccessManager *manager = new QNetworkAccessManager
360     ;
361     connect(manager, SIGNAL(finished(QNetworkReply *)), this,
362             SLOT(on_locationImageDownloaded(QNetworkReply *)))
363     ;
364     QString locationURL("http://maps.googleapis.com/maps/api/
365                           staticmap?center=");
366     locationURL.append(QString::number(latitude));
367     locationURL.append(",");
368     locationURL.append(QString::number(longitude));
369     locationURL.append("&zoom=17&size=600x750&maptype=hybrid&&
370                           format=png&visual_"
371                           "refresh=true&markers=size:mid%7Ccolor
372                           :0xff0000%7Clabel:S%
373                           "7C");
374     locationURL.append(QString::number(latitude));
375     locationURL.append(",");
376     locationURL.append(QString::number(longitude));
377     qDebug() << locationURL;
378     QUrl googleStaticMapUrl(locationURL);
379     manager->get(QNetworkRequest(googleStaticMapUrl));
380 }
381
382 void QReportGenerator::on_locationImageDownloaded(
383     QNetworkReply *reply) {
384     if (mapLocation == nullptr) {
385         mapLocation = new QImage;
386     }
387     mapLocation->loadFromData(reply->readAll());
388     if (mapLocation->isNull()) {
389         mapLocation->load("Maps/SampleLocation.png");
390     }
391
392     QTextBlock location = Report->findBlockByNumber(15);
393     QTextCursor insertMap(location);
394     insertMap.setBlockFormat(ImageGraphFormat);
395     insertMap.insertImage(*mapLocation);
396     insertMap.insertBlock();
397     insertMap.insertHtml("<br>");
398 }
399
400 QReportGenerator::~QReportGenerator()
401 {
402     delete CIElabPlot;
403     delete mapLocation;
404     delete ui;
405 }
406
407 void QReportGenerator::on_actionSave_triggered() {
408     QString fn = QFileDialog::getSaveFileName(
409         this, tr("Save Report"), QString::fromStdString(
410             Settings->SampleFolder),
411         tr("Report (*.odf)"));

```

```

405     if (!fn.isEmpty()) {
406         if (!fn.contains(tr(".odf"))) {
407             fn.append(tr(".odf"));
408         }
409         QTextDocumentWriter m_write;
410         m_write.setFileName(fn);
411         m_write.setFormat("odf");
412         m_write.write(Report);
413     }
414 }
415
416 void QReportGenerator::on_actionExport_to_PDF_triggered() {
417     QString fn = QFileDialog::getSaveFileName(
418         this, tr("Save Report"), QString::fromStdString(
419             Settings->SampleFolder),
420             tr("Report (*.pdf)"));
421     if (!fn.isEmpty()) {
422         if (!fn.contains(tr(".pdf"))) {
423             fn.append(tr(".pdf"));
424         }
425         QPrinter printer;
426         printer.setOutputFormat(QPrinter::PdfFormat);
427         printer.setOutputFileName(fn);
428         Report->print(&printer);
429     }
430 }
431
432 void QReportGenerator::SetupCIELabPPlot() {
433     if (CIELabPlot == nullptr) {
434         CIELabPlot = new QCustomPlot();
435     }
436     QPen binPen;
437     binPen.setColor(QColor("blue"));
438     binPen.setStyle(Qt::SolidLine);
439     binPen.setWidthF(1);
440
441     // Setup the CIELabplot plot
442     QCPPlotTitle *CIEtitle = new QCPPlotTitle(CIELabPlot);
443     CIEtitle->setText("mean CIE Lab - a* vs. b*");
444     CIEtitle->setFont(QFont("sans", 8, QFont::Bold));
445     CIELabPlot->plotLayout()->insertRow(0);
446     CIELabPlot->plotLayout()->addElement(0, 0, CIEtitle);
447
448     CIELabPlot->addGraph(CIELabPlot->xAxis, CIELabPlot->yAxis)
449         ;
450     CIELabPlot->graph(0)
451         ->setScatterStyle(QCPScatterStyle(QCPScatterStyle::
452             ssCircle, 8));
453     CIELabPlot->graph(0)->setPen(binPen);
454     CIELabPlot->graph(0)->setName("a* vs. b*");
455     CIELabPlot->graph(0)->setData(*Sample->GetCIELab_bVector()
456         , *Sample->GetCIELab_aVector());
457     CIELabPlot->graph(0)->setScatterStyle(QCPScatterStyle::
458             ssCross);
459     CIELabPlot->graph(0)->setLineStyle(QCPGraph::lsNone);

```

```
456
457     CIELabPlot->xAxis->setLabel("mean chromatic b*");
458     CIELabPlot->xAxis->setTickLabelFont(QFont("sans", 8, QFont
459         ::Normal));
460     CIELabPlot->xAxis->setScaleType(QCPAxis::stLinear);
461     CIELabPlot->xAxis->setRange(-128,128);
462
463     CIELabPlot->yAxis->setLabel("mean chromatic a*");
464     CIELabPlot->yAxis->setTickLabelFont(QFont("sans", 8, QFont
465         ::Normal));
466     CIELabPlot->yAxis->setScaleType(QCPAxis::stLinear);
467     CIELabPlot->yAxis->setRange(-128,128);
468     CIELabPlot->replot();
469 }
```



N. Vision Soil Analyzer Program

N.0.27 General project files

```
1  #
2  #
3  # Project created by QtCreator 2015-08-07T16:50:24
4  #
5  #
6  #
7
8  QT      += core gui concurrent
9  QMAKE_CXXFLAGS += -std=c++11
10
11 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport
12     multimedia multimediawidgets
13
14 TARGET = VSA
15 TEMPLATE = app
16 VERSION = 0.9.7
17
18 unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../../../build/install/
19
20 CONFIG(release, debug|release):DEFINES += QT_NO_DEBUG_OUTPUT
21
22
23 SOURCES += main.cpp \
24             vsemainwindow.cpp \
25             dialogsettings.cpp \
26             dialognn.cpp
27
28 HEADERS  += vsemainwindow.h \
29             dialogsettings.h \
30             dialognn.h
```

```
31
32 FORMS     += vsemainwindow.ui \
33     dialogsettings.ui \
34     dialognn.ui
35
36 #opencv
37 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui -
38     -lopencv_imgcodecs
39 INCLUDEPATH += /usr/local/include/opencv
40 INCLUDEPATH += /usr/local/include
41
42 #boost
43 DEFINES += BOOST_ALL_DYN_LINK
44 INCLUDEPATH += /usr/include/boost
45 LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_filesystem -
46     -lboost_serialization -lboost_system -lboost_iostreams
47
48 #SoilMath lib
49 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
50 INCLUDEPATH += $$PWD/../../SoilMath
51 DEPENDPATH += $$PWD/../../SoilMath
52
53 #SoilHardware lib
54 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
55     -lSoilHardware
56 INCLUDEPATH += $$PWD/../../SoilHardware
57 DEPENDPATH += $$PWD/../../SoilHardware
58
59 #SoilVision lib
60 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
61     -lSoilVision
62 INCLUDEPATH += $$PWD/../../SoilVision
63 DEPENDPATH += $$PWD/../../SoilVision
64
65 #QCustomplot lib
66 DEFINES += QCUSTOMPLOT_USE_LIBRARY
67 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
68     -lqcustomplot
69 INCLUDEPATH += $$PWD/../../qcustomplot
70 DEPENDPATH += $$PWD/../../qcustomplot
71
72 #QParticleSelector
73 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
74     -lQParticleSelector
75 INCLUDEPATH += $$PWD/../../QParticleSelector
76 DEPENDPATH += $$PWD/../../QParticleSelector
77
78 #QParticleDisplay
79 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
80     -lQParticleDisplay
81 INCLUDEPATH += $$PWD/../../QParticleDisplay
82 DEPENDPATH += $$PWD/../../QParticleDisplay
83
84 #QOpenCVQT
85 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lQOpenCVQT
86 INCLUDEPATH += $$PWD/../../QOpenCVQT
```

```

80  DEPENDPATH += $$PWD/../../QOpenCVQT
81
82 #QSoilAnalyzer
83 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilAnalyzer
84 INCLUDEPATH += $$PWD/../../SoilAnalyzer
85 DEPENDPATH += $$PWD/../../SoilAnalyzer
86
87 #QReportGenerator
88 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lQReportGenerator
89 INCLUDEPATH += $$PWD/../../QReportGenerator
90 DEPENDPATH += $$PWD/../../QReportGenerator
91
92 #NeuralNetFiles
93 NNtarget.path += $$OUT_PWD/NeuralNet
94 NNtarget.files += $$PWD/NeuralNet/*.NN
95 INSTALLS += NNtarget
96 bNNtarget.path += $$PWD/../../build/install/NeuralNet
97 bNNtarget.files += $$PWD/NeuralNet/*.NN
98 INSTALLS += bNNtarget
99
100 #SettingFiles
101 INITtarget.path += $$OUT_PWD/Settings
102 INITtarget.files += $$PWD/Settings/*.ini
103 INSTALLS += INITtarget
104 bINITtarget.path += $$PWD/../../build/install/Settings
105 bINITtarget.files += $$PWD/Settings/*.ini
106 INSTALLS += bINITtarget
107
108 #SoilSamples
109 IMGTtarget.path += $$OUT_PWD/SoilSamples
110 IMGTtarget.files += $$PWD/SoilSamples/*.VSA
111 INSTALLS += IMGTtarget
112 bIMGTtarget.path += $$PWD/../../build/install/SoilSamples
113 bIMGTtarget.files += $$PWD/SoilSamples/*.VSA
114 INSTALLS += bIMGTtarget
115
116 #Images
117 Imgttarget.path += $$OUT_PWD/Images
118 Imgttarget.files += $$PWD/Images/*
119 INSTALLS += Imgttarget
120 bImgtarget.path += $$PWD/../../build/install/Images
121 bImgtarget.files += $$PWD/Images/*
122 INSTALLS += bImgtarget
123
124 #TestedSample
125 TestedSamplesTarget.path += $$OUT_PWD/TestedSamples
126 TestedSamplesTarget.files += $$PWD/TestedSamples/*
127 INSTALLS += Imgttarget
128 bTestedSamplesTarget.path += $$PWD/../../build/install/TestedSamples
129 bTestedSamplesTarget.files += $$PWD/TestedSamples/*
130 INSTALLS += bImgtarget
131
132 RESOURCES += \

```

```
133     vsa_resources.qrc
134
135 #MainProg
136 unix {
137     target.path = $PWD/../../../../build/install
138     INSTALLS += target
139 }
140
141 DISTFILES += \
142     Settings/Default.ini \
143     NeuralNet/Default.NN \
144     Settings/User.ini \
145     SoilSamples/Eurogrit_B3_01__Cat.VSA \
146     SoilSamples/Gran_K1_0.5_2.5__01_Cat.VSA \
147     TestedSamples/Filterzand_0.2_1.6.csv \
148     TestedSamples/Magro_dol.csv \
149     TestedSamples/Gran_K1.csv \
150     TestedSamples/GL70.csv \
151     TestedSamples/Gannet_20_40.csv \
152     TestedSamples/Eurogrit.csv \
153     TestedSamples/0.8_1.25.csv


---


1 #include "vsamainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     VSAMainWindow w;
8     w.show();
9
10    return a.exec();
11 }
```

N.0.28 Main window Class

```

1 #ifndef VSAMAINWINDOW_H
2 #define VSAMAINWINDOW_H
3
4 #include <QDebug>
5 #include <QMainWindow>
6 #include <QErrorMessage>
7 #include <QMessageBox>
8 #include <QProgressBar>
9 #include <QBrush>
10
11 #include <stdint.h>
12
13 #include <qcustomplot.h>
14
15 #include "soilanalyzer.h"
16 #include "Hardware.h"
17
18 #include "dialognn.h"
19 #include "dialogsettings.h"
20 #include "qparticleselector.h"
21 #include "qreportgenerator.h"
22
23 namespace Ui {
24 class VSAMainWindow;
25 }
26
27 class VSAMainWindow : public QMainWindow {
28     Q_OBJECT
29
30 public:
31     explicit VSAMainWindow(QWidget *parent = 0);
32     ~VSAMainWindow();
33
34 private slots:
35     void on_actionSettings_triggered();
36
37     void on_analyzer_finished();
38
39     void on_actionNeuralNet_triggered();
40
41     void on_actionNewSample_triggered();
42
43     void on_actionSaveSample_triggered();
44
45     void on_actionLoadSample_triggered();
46
47     void on_actionUseLearning_toggled(bool arg1);
48
49     void on_actionCalibrate_triggered();
50
51     void on_Classification_changed(int newValue);
52
53     void on_particle_deleted();
54

```

```

105 #endif // VSAMAINWINDOW_H

1 #include "vsamainwindow.h"
2 #include "ui_vsamainwindow.h"
3
4 VSAMainWindow::VSAMainWindow(QWidget *parent)
5     : QMainWindow(parent), ui(new Ui::VSAMainWindow) {
6     ui->setupUi(this);
7
8     // Load the usersettings
9     Settings = new SoilAnalyzer::SoilSettings;
10    Settings->LoadSettings("Settings/User.ini");
11
12    // Set the message windows
13    CamError = new QErrorMessage(this);
14    SaveMeMessage = new QMessageBox(this);
15    SaveMeMessage->setText(tr("Sample is not saved, Save
16        sample?"));
17    SaveMeMessage-> addButton(QMessageBox::Abort);
18    SaveMeMessage-> addButton(QMessageBox::Close);
19
20    BacklightMessage = new QMessageBox(this);
21    BacklightMessage->setText("Turn off Frontlight! Turn on
22        Backlight!");
23    ShakeItBabyMessage = new QMessageBox(this);
24
25    // Load the Microscope
26    Microscope = new Hardware::Microscope;
27    try {
28        Microscope->FindCam(Settings->defaultWebcam)->
29            SelectedResolution =
30                &Microscope->FindCam(Settings->defaultWebcam)
31                    ->Resolutions[Settings->selectedResolution];
32    } catch (exception &e) {
33        Microscope->FindCam(0)->SelectedResolution =
34            &Microscope->FindCam(0)->Resolutions[Settings->
35                selectedResolution];
36    }
37    try {
38        if (!Microscope->openCam(Settings->defaultWebcam)) {
39            int defaultCam = 0;
40            Microscope->openCam(defaultCam);
41            Settings->defaultWebcam = Microscope->SelectedCam->
42                Name;
43        }
44    } catch (Hardware::Exception::MicroscopeException &e) {
45        if (*e.id() == EXCEPTION_OPENCAM_NR) {
46            try {
47                int defaultCam = 0;
48                Microscope->openCam(defaultCam);
49                Settings->defaultWebcam = Microscope->SelectedCam->
50                    Name;
51            } catch (Hardware::Exception::MicroscopeException &e)
52            {
53                if (*e.id() == EXCEPTION_NOCAMS_NR) {
54                    CamError->showMessage(
55

```

```

48         tr("No cams found! Connect the cam and set the
49             default"));
50         settingsWindow = new DialogSettings(this, Settings
51             , Microscope);
52     }
53 }
54
55 // Setup the sample
56 Sample = new SoilAnalyzer::Sample;
57 Images = new SoilAnalyzer::Analyzer::Images_t;
58 Analyzer = new SoilAnalyzer::Analyzer(Images, Sample,
59     Settings);
60
61 // Setup the setting Window
62 if (settingsWindow == nullptr) {
63     settingsWindow =
64         new DialogSettings(this, Settings, Microscope, &
65             Analyzer->NeuralNet);
66 }
67
68 // Setup the NN window
69 if (nnWindow == nullptr) {
70     nnWindow =
71         new DialogNN(this, &Analyzer->NeuralNet, Settings,
72             settingsWindow);
73 }
74
75 // Setup the progresbar and connect it to the Analyzer
76 Progress = new QProgressBar(ui->statusBar);
77 Progress->setMaximum(Analyzer->MaxProgress);
78 Progress->setValue(0);
79 Progress->setAlignment(Qt::AlignLeft);
80 Progress->setMinimumSize(750, 19);
81 ui->statusBar->addWidget(Progress);
82 connect(Analyzer, SIGNAL(on_progressUpdate(int)), Progress
83
84         ,
85             SLOT(setValue(int)));
86 connect(Analyzer, SIGNAL(on_progressUpdate(int)), Progress
87
88         ,
89             SLOT(setMaximum(int)));
90 connect(Analyzer, SIGNAL(on_AnalysisFinished()), this,
91             SLOT(on_analyzer_finished()));
92
93 // Setup the plot linestyles;
94 QPen pdfPen;
95 pdfPen.setColor(QColor("gray"));
96 pdfPen.setStyle(Qt::DashDotDotLine);
97 pdfPen.setWidthF(1);
98
99 QPen meanPen;
100 meanPen.setColor(QColor("darkBlue"));
101 meanPen.setStyle(Qt::DashLine);
102 meanPen.setWidthF(1);
103
104 QPen binPen;

```

```

97     binPen.setColor((QColor("blue")));
98     binPen.setStyle(Qt::SolidLine);
99     binPen.setWidthF(2);
100
101 // Setup the PSD plot
102 QCPPPlotTitle *PSDtitle = new QCPPPlotTitle(ui->Qplot_PSD);
103 PSDtitle->setText("Particle Size Distribution");
104 PSDtitle->setFont(QFont("sans", 8, QFont::Bold));
105 ui->Qplot_PSD->plotLayout()->insertRow(0);
106 ui->Qplot_PSD->plotLayout()->addElement(0, 0, PSDtitle);
107
108 ui->Qplot_PSD->addGraph(ui->Qplot_PSD->xAxis, ui->
109     Qplot_PSD->yAxis);
110 ui->Qplot_PSD->graph(0)
111     ->setScatterStyle(QCPSscatterStyle(QCPSscatterStyle:::
112         ssCircle, 8));
113 ui->Qplot_PSD->graph(0)->setPen(binPen);
114 ui->Qplot_PSD->graph(0)->setName("Particle Size
115     Distribution");
116 ui->Qplot_PSD->graph(0)->addToLegend();
117
118 ui->Qplot_PSD->xAxis->setLabel("Particle size [mm]");
119 ui->Qplot_PSD->xAxis->setRange(0.01, 10);
120 ui->Qplot_PSD->xAxis->setAutoTicks(false);
121 ui->Qplot_PSD->xAxis->setTickVector(QVector<double>:::
122     fromStdVector(PSDTicks));
123 ui->Qplot_PSD->xAxis->setTickLabelRotation(30);
124 ui->Qplot_PSD->xAxis->setTickLabelFont(QFont("sans", 8,
125     QFont::Normal));
126 ui->Qplot_PSD->xAxis->setScaleType(QCPAxis::stLogarithmic)
127     ;
128
129 QFont legendfont;
130 legendfont.setPointSize(10);
131 ui->Qplot_PSD->legend->setFont(legendfont);
132 ui->Qplot_PSD->legend->setSelectedFont(legendfont);
133 ui->Qplot_PSD->legend->setVisible(true);
134 ui->Qplot_PSD->axisRect()->insetLayout()->
135     setInsetAlignment(
136         0, Qt::AlignTop | Qt::AlignLeft);
137
138 ui->Qplot_PSD->yAxis->setLabel("Percentage [%]");
139 ui->Qplot_PSD->yAxis->setRange(0, 100);
140 ui->Qplot_PSD->setInteractions(QCP::iRangeDrag | QCP:::
141         iRangeZoom);
142 ui->Qplot_PSD->yAxis->grid()->setSubGridVisible(true);
143
144 connect(ui->Qplot_PSD, SIGNAL(mouseDoubleClick(QMouseEvent
145         *)), this,
146             SLOT(on_reset_graph(QMouseEvent *)));
147 ui->Qplot_PSD->setContextMenuPolicy(Qt::CustomContextMenu)
148     ;
149 connect(ui->Qplot_PSD, SIGNAL(customContextMenuRequested(
150         QPoint)), this,
151             SLOT(on_PSD_contextMenuRequest(QPoint)));

```

```

142 // Setup the Roundness plot
143 QCPPlotTitle *Roundnesstitle = new QCPPlotTitle(ui->
144     QPlot_Roudness);
145 Roundnesstitle->setText("Sphericity Histogram");
146 Roundnesstitle->setFont(QFont("sans", 8, QFont::Bold));
147 ui->QPlot_Roudness->plotLayout()->insertRow(0);
148 ui->QPlot_Roudness->plotLayout()->addElement(0, 0,
149     Roundnesstitle);
150
151 ui->QPlot_Roudness->addGraph(ui->QPlot_Roudness->xAxis,
152                                     ui->QPlot_Roudness->yAxis2);
153 ui->QPlot_Roudness->addGraph(ui->QPlot_Roudness->xAxis,
154                                     ui->QPlot_Roudness->yAxis2);
155 ui->QPlot_Roudness->graph(0)->setPen(pdfPen);
156 ui->QPlot_Roudness->graph(1)->setPen(meanPen);
157
158 RoundnessBars =
159     new QCPBars(ui->QPlot_Roudness->xAxis, ui->
160                 QPlot_Roudness->yAxis);
161 ui->QPlot_Roudness->addPlottable(RoundnessBars);
162 RoundnessBars->setPen(binPen);
163
164 ui->QPlot_Roudness->xAxis->setAutoTicks(false);
165 ui->QPlot_Roudness->xAxis->setAutoTickLabels(false);
166 ui->QPlot_Roudness->xAxis->setTickVector(
167     QVector<double>::fromStdVector(RoundnessTicks));
168 ui->QPlot_Roudness->xAxis->setTickVectorLabels(
169     RoundnessCat);
170 ui->QPlot_Roudness->xAxis->setTickLabelRotation(30);
171 ui->QPlot_Roudness->xAxis->setSubTickCount(0);
172 ui->QPlot_Roudness->xAxis->setTickLength(0, 4);
173 ui->QPlot_Roudness->xAxis->grid()->setVisible(true);
174 ui->QPlot_Roudness->xAxis->setRange(0, 4);
175 ui->QPlot_Roudness->xAxis->setLabel("Count [-]");
176 ui->QPlot_Roudness->xAxis->setLabelFont(QFont("sans", 8,
177     QFont::Bold));
178 ui->QPlot_Roudness->xAxis->setTickLabelFont(QFont("sans",
179     8, QFont::Normal));
180 ui->QPlot_Roudness->xAxis->setPadding(25);
181 ui->QPlot_Roudness->yAxis->setLabel("Sphericity [-]");
182 ui->QPlot_Roudness->yAxis->setLabelFont(QFont("sans", 8,
183     QFont::Bold));
184
185 // Setup the angularity plot
186 QCPPlotTitle *Angularitytitle = new QCPPlotTitle(ui->
187     QPlot_Angularity);
188 Angularitytitle->setText("Angularity Histogram");
189 Angularitytitle->setFont(QFont("sans", 8, QFont::Bold));
190 ui->QPlot_Angularity->plotLayout()->insertRow(0);
191 ui->QPlot_Angularity->plotLayout()->addElement(0, 0,
192     Angularitytitle);
193
194 ui->QPlot_Angularity->addGraph(ui->QPlot_Angularity->xAxis,
195                                     ui->QPlot_Angularity->
196                                     yAxis2);

```

```

187     ui->QPlot_Angularity->addGraph(ui->QPlot_Angularity->xAxis
188             ,
189             ui->QPlot_Angularity->
190             yAxis2);
191     AngularityBars =
192         new QCPlotBars(ui->QPlot_Angularity->xAxis, ui->
193             QPlot_Angularity->yAxis);
194     ui->QPlot_Angularity->addPlottable(AngularityBars);
195     AngularityBars->setPen(binPen);
196
197     ui->QPlot_Angularity->xAxis->setAutoTicks(false);
198     ui->QPlot_Angularity->xAxis->setAutoTickLabels(false);
199     ui->QPlot_Angularity->xAxis->setTickVector(
200         QVector<double>::fromStdVector(AngularityTicks));
201     ui->QPlot_Angularity->xAxis->setTickVectorLabels(
202         AngularityCat);
203     ui->QPlot_Angularity->xAxis->setTickLabelRotation(30);
204     ui->QPlot_Angularity->xAxis->setSubTickCount(0);
205     ui->QPlot_Angularity->xAxis->setTickLength(0, 4);
206     ui->QPlot_Angularity->xAxis->grid()->setVisible(true);
207     ui->QPlot_Angularity->xAxis->setRange(0, 7);
208     ui->QPlot_Angularity->xAxis->setLabel("Count [-]");
209     ui->QPlot_Angularity->xAxis->setLabelFont(QFont("sans", 8,
210         QFont::Bold));
211     ui->QPlot_Angularity->xAxis->setTickLabelFont(
212         QFont("sans", 8, QFont::Normal));
213     ui->QPlot_Angularity->yAxis->setLabel("Sphericity [-]");
214     ui->QPlot_Angularity->yAxis->setLabelFont(QFont("sans", 8,
215         QFont::Bold));
216     ui->QPlot_Angularity->graph(0)->setPen(pdfPen);
217     ui->QPlot_Angularity->graph(1)->setPen(meanPen);
218
219     // Setup the Amplitude diagram
220     QCPlotTitle *Amptitle = new QCPlotTitle(ui->QPlot_Amp);
221     Amptitle->setText("Fast Fourier Amplitude for the current
222         particle");
223     Amptitle->setFont(QFont("sans", 8, QFont::Bold));
224     ui->QPlot_Amp->plotLayout()->insertRow(0);
225     ui->QPlot_Amp->plotLayout()->addElement(0, 0, Amptitle);
226
227     ui->QPlot_Amp->addGraph(ui->QPlot_Amp->xAxis, ui->
228         QPlot_Amp->yAxis);
229
230     ui->QPlot_Amp->xAxis->setTickLabelRotation(30);
231     ui->QPlot_Amp->xAxis->setSubTickCount(0);
232     ui->QPlot_Amp->xAxis->setTickLength(0, 4);
233     ui->QPlot_Amp->xAxis->grid()->setVisible(true);
234     ui->QPlot_Amp->xAxis->setRange(0, 512);
235     ui->QPlot_Amp->xAxis->setLabel("Frequency [-]");
236     ui->QPlot_Amp->xAxis->setLabelFont(QFont("sans", 8, QFont
237         ::Bold));
238     ui->QPlot_Amp->xAxis->setTickLabelFont(QFont("sans", 8,
239         QFont::Normal));
240     ui->QPlot_Amp->yAxis->setLabel("Amplitude [-]");
241     ui->QPlot_Amp->yAxis->setLabelFont(QFont("sans", 8, QFont
242         ::Bold));

```

```

232     ui->QPlot_Amp->yAxis->setScaleType(QCPAxis::stLogarithmic)
233     ;
234     ui->QPlot_Amp->graph()->setPen(binPen);
235     ui->QPlot_Amp->graph()->setLineStyle(QCPGraph::lsLine);
236     ui->QPlot_Amp->graph()->setBrush(QBrush(QColor
237         (50,50,200,40)));
238
239     // Connect the Particle display and Selector
240     connect(ui->widget_ParticleSelector, SIGNAL(valueChanged(
241         int)), this,
242             SLOT(on_Classification_changed(int)));
243     connect(ui->widget_ParticleDisplay, SIGNAL(
244         shapeClassificationChanged(int)),
245             ui->widget_ParticleSelector, SLOT(setValue(int)));
246     connect(ui->widget_ParticleDisplay, SIGNAL(particleDeleted
247         ()), this,
248             SLOT(on_particle_deleted()));
249     connect(ui->widget_ParticleDisplay, SIGNAL(particleChanged
250         (int)), this,
251             SLOT(on_particleChanged(int)));
252 }
253
254 VSAMainWindow::~VSAMainWindow() {
255     delete Settings;
256     delete Microscope;
257     delete Analyzer;
258     delete Sample;
259     delete Images;
260
261     delete settingsWindow;
262     delete nnWindow;
263     delete CamError;
264     delete SaveMeMessage;
265     delete BacklightMessage;
266     delete ShakeItBabyMessage;
267     delete ui;
268 }
269
270 void VSAMainWindow::on_actionSettings_triggered() {
271     settingsWindow->openTab(0);
272     settingsWindow->show();
273 }
274
275 void VSAMainWindow::on_analyzer_finished() {
276     if (!ParticleDisplayerFilled && Sample->ParticlePopulation
277         .size() > 0) {
278         ui->widget_ParticleDisplay->SetSample(Sample);
279     }
280     SetPSDgraph();

```

```

280     setRoundnessHistogram();
281     setAngularityHistogram();
282     ParticleDisplayerFilled = true;
283 }
284
285 void VSAMainWindow::SetPSDgraph() {
286     std::vector<double> stdPSDvalue(Sample->PSD.CFD, Sample->
287         PSD.CFD + 15);
288     ui->Qplot_PSD->graph(0)->setData(PSDTicks, stdPSDvalue);
289     ui->Qplot_PSD->replot();
290 }
291
292 void VSAMainWindow::setRoundnessHistogram() {
293     // Setup the Histogram bins
294     std::vector<double> stdValues(Sample->Roundness.bins + 1,
295                                     Sample->Roundness.bins + 4);
296
297     ui->QPlot_Roudness->yAxis->setRange(
298         0, static_cast<double>(Sample->Roundness.
299             HighestFrequency())));
300     RoundnessBars->setData(RoundnessTicks, stdValues);
301
302     // Setup the Prediction Density Function
303     std::vector<double> stdPDFkey, stdPDFvalues;
304     Sample->Roundness.GetPDFfunction(stdPDFkey, stdPDFvalues,
305         0.2, 0, 4);
306     ui->QPlot_Roudness->graph(0)->setData(stdPDFkey,
307         stdPDFvalues);
308     ui->QPlot_Roudness->yAxis2->setRange(0, Sample->Roundness.
309         HighestPDF);
310
311     // Setup the mean Vector
312     QVector<double> meanKey(2, static_cast<double>(Sample->
313         Roundness.Mean));
314     QVector<double> meanValue(2);
315     meanValue[0] = 0;
316     meanValue[1] = Sample->Roundness.HighestPDF;
317     ui->QPlot_Roudness->graph(1)->setData(meanKey, meanValue);
318     ui->QPlot_Roudness->replot();
319 }
320
321 void VSAMainWindow::setAngularityHistogram() {
322     // Setup the Histogram bins
323     std::vector<double> stdValues(Sample->Angularity.bins + 1,
324                                     Sample->Angularity.bins + 7)
325                                     ;
326
327     ui->QPlot_Angularity->yAxis->setRange(
328         0, static_cast<double>(Sample->Angularity.
329             HighestFrequency()));
330     AngularityBars->setData(AngularityTicks, stdValues);
331
332     // Setup the Prediction Density Function
333     std::vector<double> stdPDFkey, stdPDFvalues;
334     Sample->Angularity.GetPDFfunction(stdPDFkey, stdPDFvalues,
335         0.2, 0, 7);

```

```

327     ui->QPlot_Angularity->graph(0)->setData(stdPDFkey ,
328         stdPDFvalues);
329     ui->QPlot_Angularity->yAxis2->setRange(0, Sample->
330         Angularity.HighestPDF);
331     // Setup the mean Vector
332     QVector<double> meanKey(2, static_cast<double>(Sample->
333         Angularity.Mean));
334     QVector<double> meanValue(2);
335     meanValue[0] = 0;
336     meanValue[1] = Sample->Angularity.HighestPDF;
337     ui->QPlot_Angularity->graph(1)->setData(meanKey, meanValue
338         );
339     ui->QPlot_Angularity->replot();
340 }
341
342 void VSAMainWindow::setAmpgraph() {
343     ui->QPlot_Amp->graph(0)->clearData();
344     ComplexVect_t *comp =
345         &ui->widget_ParticleDisplay->SelectedParticle->
346             FFDescriptors;
347     uint32_t count = (comp->size() > 64) ? 64 : comp->size();
348     for (uint32_t i = 0; i < count; i++) {
349         ui->QPlot_Amp->graph(0)->addData(i, abs(comp->at(i)));
350     }
351     ui->QPlot_Amp->rescaleAxes();
352     ui->QPlot_Amp->replot();
353 }
354
355 void VSAMainWindow::on_particleChanged(int newPart) {
356     setAmpgraph(); }
357
358 void VSAMainWindow::on_actionNeuralNet_triggered() {
359     if (nnWindow != nullptr) {
360         nnWindow =
361             new DialogNN(this, &Analyzer->NeuralNet, Settings,
362                         settingsWindow);
363     }
364     nnWindow->show();
365 }
366
367 void VSAMainWindow::on_actionNewSample_triggered() {
368     if (Sample->ChangesSinceLastSave) {
369         if (SaveMeMessage->exec() == QMessageBox::Abort) {
370             return;
371         }
372         delete Sample;
373         Sample = nullptr;
374         delete Images;
375         Images = nullptr;
376         Sample = new SoilAnalyzer::Sample;
377         Images = new SoilAnalyzer::Analyzer::Images_t;
378         TakeSnapShots();
379         try {
380             Analyzer->Analyse(Images, Sample, Settings);
381         }
382     }
383 }

```

```

376     } catch (SoilAnalyzer::Exception::SoilAnalyzerException &e
377         ) {
378     if (*e.id() == EXCEPTION_NO_SNAPSHOTS_NR) {
379         CamError->showMessage(
380             "No images acquired! Check your microscope settings
381             ");
382     }
383     Sample->ChangesSinceLastSave = true;
384     if (Sample->ParticlePopulation.size() > 0) {
385         ui->widget_ParticleSelector->setDisabled(
386             false,
387             ui->widget_ParticleDisplay->SelectedParticle->
388                 Classification.Category);
389     }
390 }
391 void VSAMainWindow::TakeSnapShots() {
392     Analyzer->SIfactorDet = true; // remember to remove
393     if (!Analyzer->SIfactorDet) {
394         QMessageBox *DetSIFactor = new QMessageBox(this);
395         DetSIFactor->setText("Put calibration Disc under the
396             microscope");
397         DetSIFactor->exec();
398         on_actionCalibrate_triggered();
399         DetSIFactor->setText("Place sample under the microscope"
400             );
401         DetSIFactor->exec();
402     }
403     if (Settings->useBacklightProjection && !Settings->useHDR)
404     {
405         for (uint32_t i = 0; i < Settings->StandardNumberOfShots
406             ; i++) {
407             SoilAnalyzer::Analyzer::Image_t newShot;
408             newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
409             Microscope->GetFrame(newShot.FrontLight);
410             BacklightMessage->exec();
411             Microscope->GetFrame(newShot.BackLight);
412             Images->push_back(newShot);
413             QString ShakeMsg = "Shake it baby! ";
414             int number = Settings->StandardNumberOfShots - i;
415             ShakeMsg.append(QString::number(number));
416             ShakeMsg.append(" to go!");
417             ShakeItBabyMessage->setText(ShakeMsg);
418             ShakeItBabyMessage->exec();
419     }
420 } else if (Settings->useBacklightProjection && Settings->
421             useHDR) {
422     for (uint32_t i = 0; i < Settings->StandardNumberOfShots
423             ; i++) {
424         SoilAnalyzer::Analyzer::Image_t newShot;
425         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
426         Microscope->GetHDRFrame(newShot.FrontLight, Settings->
427             HDRframes);
428         BacklightMessage->exec();
429 }
```

```

422     Microscope->GetFrame(newShot.BackLight);
423     Images->push_back(newShot);
424     QString ShakeMsg = "Shake it baby! ";
425     int number = Settings->StandardNumberOfShots - i - 1;
426     ShakeMsg.append(QString::number(number));
427     ShakeMsg.append(" to go!");
428     ShakeItBabyMessage->setText(ShakeMsg);
429     ShakeItBabyMessage->exec();
430 }
431 } else if (!Settings->useBacklightProjection && Settings->
432     useHDR) {
433     for (uint32_t i = 0; i < Settings->StandardNumberOfShots
434         ; i++) {
435         SoilAnalyzer::Analyzer::Image_t newShot;
436         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
437         Microscope->GetHDRFrame(newShot.FrontLight, Settings->
438             HDRframes);
439         Images->push_back(newShot);
440         QString ShakeMsg = "Shake it baby! ";
441         int number = Settings->StandardNumberOfShots - i - 1;
442         ShakeMsg.append(QString::number(number));
443         ShakeMsg.append(" to go!");
444         ShakeItBabyMessage->setText(ShakeMsg);
445         ShakeItBabyMessage->exec();
446     }
447 } else if (!Settings->useBacklightProjection && !Settings
448     ->useHDR) {
449     for (uint32_t i = 0; i < Settings->StandardNumberOfShots
450         ; i++) {
451         SoilAnalyzer::Analyzer::Image_t newShot;
452         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
453         Microscope->GetFrame(newShot.FrontLight);
454         Images->push_back(newShot);
455         QString ShakeMsg = "Shake it baby! ";
456         int number = Settings->StandardNumberOfShots - i - 1;
457         ShakeMsg.append(QString::number(number));
458         ShakeMsg.append(" to go!");
459         ShakeItBabyMessage->setText(ShakeMsg);
460         ShakeItBabyMessage->exec();
461     }
462 }
463 void VSAMainWindow::on_actionSaveSample_triggered() {
464     QString fn = QFileDialog::getSaveFileName(
465         this, tr("Save Sample"), QString::fromStdString(
466             Settings->SampleFolder),
467             tr("Sample (*.VSA)"));
468     if (!fn.isEmpty()) {
469         if (!fn.contains(tr(".VSA"))) {
470             fn.append(tr(".VSA"));
471         }
472         Sample->IsLoadedFromDisk = true;
473         Sample->ChangesSinceLastSave = false;
474         Sample->Save(fn.toStdString());
475         qDebug() << "Saving finished";

```

```

472     }
473 }
474
475 void VSAMainWindow::on_actionLoadSample_triggered() {
476     if (Sample->ChangesSinceLastSave) {
477         if (SaveMeMessage->exec() == QMessageBox::Abort) {
478             return;
479         }
480     }
481
482     QString fn = QFileDialog::getOpenFileName(
483         this, tr("Open Sample"), QString::fromStdString(
484             Settings->SampleFolder),
485         tr("Sample (*.VSA)"));
486     if (!fn.isEmpty()) {
487         if (!fn.contains(tr(".VSA")))
488             fn.append(tr(".VSA"));
489
490         delete Sample;
491         Sample = nullptr;
492         delete Images;
493         Images = nullptr;
494         Sample = new SoilAnalyzer::Sample;
495         Images = new SoilAnalyzer::Analyzer::Images_t;
496         try {
497             Sample->Load(fn.toStdString());
498         } catch (boost::archive::archive_exception &e) {
499             // qDebug() << *e.what();
500         }
501         ParticleDisplayerFilled = false;
502         Sample->Angularity.Data = Sample->GetAngularityVector()
503             ->data();
504         Sample->Roundness.Data = Sample->GetRoundnessVector()->
505             data();
506         Sample->PSD.Data = Sample->GetPSDVector()->data();
507         Analyzer->Results = Sample;
508         on_analyzer_finished();
509         ui->widget_ParticleSelector->setDisabled(
510             false,
511             ui->widget_ParticleDisplay->SelectedParticle->
512                 Classification.Category);
513     }
514 }
515
516 void VSAMainWindow::on_actionUseLearning_toggled(bool arg1)
517 {
518     Analyzer->PredictShape = !arg1;
519 }
520
521
522 void VSAMainWindow::on_Classification_changed(int newValue)

```

```

523     {
524     uint8_t *Cat =
525         &ui->widget_ParticleDisplay->SelectedParticle->
526             Classification.Category;
527     if ((*Cat - 1) % 6 != (newValue - 1) % 6) {
528         Sample->ParticleChangedStateAngularity = true;
529     }
530     if ((*Cat - 1) / 6 != (newValue - 1) / 6) {
531         Sample->ParticleChangedStateRoundness = true;
532     }
533     ui->widget_ParticleDisplay->SelectedParticle->
534         Classification.Category =
535             newValue;
536     ui->widget_ParticleDisplay->SelectedParticle->
537         Classification.ManualSet = true;
538     Sample->ChangesSinceLastSave = true;
539     Analyzer->Analyse();
540     ui->widget_ParticleDisplay->next();
541 }
542
543 void VSAMainWindow::on_particle_deleted() { Analyzer->
544     Analyse(); }
545
546 void VSAMainWindow::on_actionAutomatic_Shape_Pediction_triggered(bool checked)
547 {
548     Settings->PredictTheShape = checked;
549 }
550
551
552 void VSAMainWindow::on_actionReport_Generator_triggered() {
553     if (ReportGenWindow == nullptr) {
554         ReportGenWindow =
555             new QReportGenerator(this, Sample, Settings, ui->
556                                 Qplot_PSD,
557                                 ui->QPlot_Roudness, ui->
558                                 QPlot_Angularity);
559     }
560     ReportGenWindow->show();
561 }
562
563 void VSAMainWindow::on_PSD_contextMenuRequest(QPoint point)
564 {
565     QMenu *menu = new QMenu(this);
566     menu->setAttribute(Qt::WA_DeleteOnClose);
567
568     menu->addAction("Compare against...", this, SLOT(
569         on_compare_against()));
570     menu->addAction("Restore", this, SLOT(on_restore_PSD()));
571 }

```

```

567     menu->popup(ui->Qplot_PSD->mapToGlobal(point));
568 }
569
570 void VSAMainWindow::on_compare_against() {
571     QString fn = QFileDialog::getOpenFileName(
572         this, tr("Open CSV"), QString::fromStdString(Settings
573             ->SampleFolder),
574         tr("Comma Separated Value (*.csv)"));
575     if (!fn.isEmpty()) {
576         if (!fn.contains(tr(".csv"))) {
577             fn.append(tr(".csv"));
578         }
579         if (ui->Qplot_PSD->graphCount() > 1) {
580             ui->Qplot_PSD->legend->removeItem(1);
581             ui->Qplot_PSD->removeGraph(1);
582         }
583
584         QStringList rows;
585         QStringList cellValues;
586
587         QFile f(fn);
588         if (f.open(QIODevice::ReadOnly)) {
589             QString data;
590             data = f.readAll();
591             rows = data.split('\n');
592             f.close();
593             for (uint32_t i = 0; i < rows.size(); i++) {
594                 QStringList cols = rows[i].split(',');
595                 for (uint32_t j = 0; j < cols.size(); j++) {
596                     cellValues.append(cols[j]);
597                 }
598             }
599             cellValues.removeLast();
600
601             std::vector<double> compValues(15);
602             for (uint32_t i = 0; i < cellValues.size(); i += 4) {
603                 bool conversionSucces = false;
604                 double binValue = cellValues[i].toDouble(&
605                     conversionSucces);
606                 qDebug() << cellValues[i + 3];
607                 if (conversionSucces) {
608                     for (uint32_t j = 0; j < 15; j++) {
609                         if (binValue == PSDTicks[j]) {
610                             compValues[j] = cellValues[i + 3].toDouble();
611                         }
612                     }
613                 }
614                 ui->Qplot_PSD->addGraph(ui->Qplot_PSD->xAxis, ui->
615                     Qplot_PSD->yAxis);
616                 ui->Qplot_PSD->graph(1)->setData(PSDTicks, compValues)
617                     ;
618                 QPen compPen;
619                 compPen.setColor(QColor("darkBlue"));
620                 compPen.setStyle(Qt::DashLine);

```

```
619     compPen.setWidthF(1);
620     ui->Qplot_PSD->graph(1)->setPen(compPen);
621     ui->Qplot_PSD->graph(1)->setName("Compared Particle
622         Size Distribution");
623     ui->Qplot_PSD->graph(1)->addToLegend();
624     ui->Qplot_PSD->replot();
625   }
626 }
627
628 void VSAMainWindow::on_restore_PSD() {
629   if (ui->Qplot_PSD->graphCount() > 1) {
630     ui->Qplot_PSD->legend->removeItem(1);
631     ui->Qplot_PSD->removeGraph(1);
632   }
633   on_reset_graph(nullptr);
634 }
```

N.0.29 Dialog window Class

```
1 #ifndef DIALOGSETTINGS_H
2 #define DIALOGSETTINGS_H
3
4 #include <QDialog>
5 #include <soilsettings.h>
6 #include <QFileDialog>
7 #include <QString>
8 #include <QDir>
9 #include <QSlider>
10 #include "Hardware.h"
11
12 namespace Ui {
13 class DialogSettings;
14 }
15
16 class DialogSettings : public QDialog {
17 Q_OBJECT
18
19 public:
20     SoilAnalyzer::SoilSettings *Settings = nullptr;
21     explicit DialogSettings(QWidget *parent = 0,
22                             SoilAnalyzer::SoilSettings *
23                             settings = nullptr,
24                             Hardware::Microscope *microscope =
25                             nullptr,
26                             SoilMath::NN *nn = nullptr, bool
27                             openNN = false);
28     ~DialogSettings();
29
30     void openTab(int newValue);
31
32     void on_pushButton_RestoreDefault_clicked();
33
34     void on_pushButton_Open_clicked();
35
36     void on_pushButton_Save_clicked();
37
38     void on_checkBox_Backlight_clicked(bool checked);
39
40     void on_comboBox_Microscopes_currentIndexChanged(const
41             QString &arg1);
42
43     void on_comboBox_Resolution_currentIndexChanged(int index)
44             ;
45
46     void on_checkBox_useHDR_clicked(bool checked);
47
48     void on_spinBox_NoFrames_editingFinished();
49
50     void on_doubleSpinBox_LightLevel_editingFinished();
51
52     void on_checkBox_useRainbow_clicked(bool checked);
```

```
50 void on_checkBox_InvertEncoder_clicked(bool checked);
51 void on_checkBox_useCUDA_clicked(bool checked);
52 void on_horizontalSlider_BrightFront_valueChanged(int
53     value);
54 void on_horizontalSlider_ContrastFront_valueChanged(int
55     value);
56 void on_horizontalSlider_SaturationFront_valueChanged(int
57     value);
58 void on_horizontalSlider_HueFront_valueChanged(int value);
59 void on_horizontalSlider_SharpnessFront_valueChanged(int
60     value);
61 void on_horizontalSlider_BrightProj_valueChanged(int value
62     );
63 void on_horizontalSlider_ContrastProj_valueChanged(int
64     value);
65 void on_horizontalSlider_SaturationProj_valueChanged(int
66     value);
67 void on_horizontalSlider_HueProj_valueChanged(int value);
68 void on_horizontalSlider_SharpnessProj_valueChanged(int
69     value);
70 void on_cb_use_adaptContrast_3_clicked(bool checked);
71 void on_cb_useBlur_3_clicked(bool checked);
72 void on_rb_useDark_3_toggled(bool checked);
73 void on_cb_ignoreBorder_3_clicked(bool checked);
74 void on_cb_fillHoles_3_clicked(bool checked);
75 void on_sb_sigmaFactor_3_editingFinished();
76 void on_rb_useOpen_3_clicked(bool checked);
77 void on_rb_useClose_3_clicked(bool checked);
78 void on_rb_useErode_3_clicked(bool checked);
79 void on_rb_useDilate_3_clicked(bool checked);
80 void on_sb_morphMask_3_editingFinished();
81 void on_spinBox_MaxGen_editingFinished();
82
```

```

98     void on_spinBox_PopSize_editingFinished();
99
100    void on_doubleSpinBox_MutationRate_editingFinished();
101
102    void on_spinBox_Elitisme_editingFinished();
103
104    void on_doubleSpinBox_endError_editingFinished();
105
106    void on_doubleSpinBox_maxWeight_editingFinished();
107
108    void on_doubleSpinBox_MinWeight_editingFinished();
109
110    void on_doubleSpinBox_Beta_editingFinished();
111
112    void on_spinBox_InputNeurons_editingFinished();
113
114    void on_spinBox_HiddenNeurons_editingFinished();
115
116    void on_spinBox_OutputNeurons_editingFinished();
117
118    void on_pushButton_selectSampleFolder_clicked();
119
120    void on_pushButton_SelectSettingFolder_clicked();
121
122    void on_pushButton_SelectNNFolder_clicked();
123
124    void on_pushButton_SelectNN_clicked();
125
126    void on_spinBox_NoShots_editingFinished();
127
128    void on_checkBox_PredictShape_clicked(bool checked);
129
130    void on_checkBox_revolt_clicked(bool checked);
131
132 private:
133     Ui::DialogSettings *ui;
134     Hardware::Microscope *Microscope;
135     SoilMath::NN *NN;
136     bool initfase = true;
137     void SetCamControl(Hardware::Microscope::Cam_t *
138                         selectedCam,
139                         QSlider *Brightness, QSlider *Contrast,
140                         QSlider *Saturation, QSlider *Hue,
141                         QSlider *Sharpness);
142
143 #endif // DIALOGSETTINGS_H


---


1 #include "dialogsettings.h"
2 #include "ui_dialogsettings.h"
3 #include <opencv2/core.hpp>
4
5 DialogSettings::DialogSettings(QWidget *parent,
6                               SoilAnalyzer::SoilSettings *
7                               settings,
8                               Hardware::Microscope *
9                               microscope);

```

```
8                     microscope,
9                     SoilMath::NN *nn, bool openNN
10                    )
11
12     : QDialog(parent), ui(new Ui::DialogSettings) {
13
14     ui->setupUi(this);
15
16     if (settings == nullptr) {
17         settings = new SoilAnalyzer::SoilSettings;
18     }
19
20     Settings = settings;
21
22     if (microscope == nullptr) {
23         microscope = new Hardware::Microscope;
24     }
25
26     if (nn == nullptr) {
27         nn = new SoilMath::NN;
28     }
29
30
31     // Setup the Hardware tab
32     Microscope = microscope;
33     QStringList Cams;
34     for (uint32_t i = 0; i < Microscope->AvailableCams.size();
35           i++) {
36         Cams << Microscope->AvailableCams[i].Name.c_str();
37     }
38     ui->comboBox_Microscopes->addItems(Cams);
39     ui->comboBox_Microscopes->setcurrentIndex(Microscope->
40         SelectedCam->ID);
41
42     QStringList Resolutions;
43     for (uint32_t i = 0; i < Microscope->SelectedCam->
44         Resolutions.size(); i++) {
45         Resolutions << Microscope->SelectedCam->Resolutions[i].
46             to_string().c_str();
47     }
48     ui->comboBox_Resolution->addItems(Resolutions);
49     ui->comboBox_Resolution->setcurrentIndex(
50         Microscope->SelectedCam->SelectedResolution->ID);
51
52     ui->spinBox_NoShots->setValue(Settings->
53         StandardNumberOfShots);
54
55     ui->spinBox_NoFrames->setValue(Settings->HDRframes);
56     ui->spinBox_NoFrames->setDisabled(true);
57     ui->label_nf->setDisabled(true);
58
59     ui->checkBox_Backlight->setChecked(Settings->
60         useBacklightProjection);
61     ui->tabWidget_Hardware->setTabEnabled(2, Settings->
62         useBacklightProjection);
63
64     ui->checkBox_InvertEncoder->setChecked(Settings->encInv);
65     ui->checkBox_useCUDA->setChecked(Settings->useCUDA);
66
67     Settings->useCUDA = false;
68     ui->checkBox_useCUDA->setDisabled(true);
69
70     ui->checkBox_useHDR->setChecked(Settings->useHDR);
```

```

55     ui->checkBox_useRainbow->setChecked(Settings->
56         enableRainbow);
57
58     // Get system info
59     struct utsname unameData;
60     uname(&unameData);
61
62     ui->label_machinename->setText(tr(unameData.machine));
63     ui->label_nodename->setText(tr(unameData.nodename));
64     ui->label_releasename->setText(tr(unameData.release));
65     ui->label_systemname->setText(tr(unameData.sysname));
66     ui->label_versionname->setText(tr(unameData.version));
67     if (Microscope->RunEnv == Hardware::Microscope::X64) {
68         ui->checkBox_useRainbow->setDisabled(true);
69         ui->checkBox_InvertEncoder->setDisabled(true);
70         ui->doubleSpinBox_LightLevel->setDisabled(true);
71         ui->label_ll->setDisabled(true);
72     }
73
74     SetCamControl(
75         Microscope->SelectedCam, ui->
76             horizontalSlider_BrightFront,
77             ui->horizontalSlider_ContrastFront, ui->
78                 horizontalSlider_SaturationFront,
79                 ui->horizontalSlider_HueFront, ui->
80                     horizontalSlider_SharpnessFront);
81     ui->horizontalSlider_BrightFront->setValue(Settings->
82         Brightness_front);
83     ui->horizontalSlider_ContrastFront->setValue(Settings->
84         Contrast_front);
85     ui->horizontalSlider_HueFront->setValue(Settings->
86         Hue_front);
87     ui->horizontalSlider_SaturationFront->setValue(Settings->
88         Saturation_front);
89     ui->horizontalSlider_SharpnessFront->setValue(Settings->
90         Sharpness_front);
91
92     SetCamControl(
93         Microscope->SelectedCam, ui->
94             horizontalSlider_BrightProj,
95             ui->horizontalSlider_ContrastProj, ui->
96                 horizontalSlider_SaturationProj,
97                 ui->horizontalSlider_HueProj, ui->
98                     horizontalSlider_SharpnessProj);
99     ui->horizontalSlider_BrightProj->setValue(Settings->
100         Brightness_proj);
101    ui->horizontalSlider_ContrastProj->setValue(Settings->
102         Contrast_proj);
103    ui->horizontalSlider_HueProj->setValue(Settings->Hue_proj)
104        ;
105    ui->horizontalSlider_SaturationProj->setValue(Settings->
106         Saturation_proj);
107    ui->horizontalSlider_SharpnessProj->setValue(Settings->
108         Sharpness_proj);
109
110    // Setup the Vision tab

```

```
94     ui->cb_fillHoles_3->setChecked(Settings->fillHoles);
95     ui->cb_ignoreBorder_3->setChecked(Settings->
96         ignorePartialBorderParticles);
97     ui->cb_useBlur_3->setChecked(Settings->useBlur);
98     if (!Settings->useBlur) {
99         ui->sb_blurMask_3->setEnabled(false);
100    }
101    ui->cb_use_adaptContrast_3->setChecked(Settings->
102        useAdaptiveContrast);
103    if (!Settings->useAdaptiveContrast) {
104        ui->sb_adaptContrastFactor_3->setEnabled(false);
105        ui->sb_adaptContrastKernel_3->setEnabled(false);
106    }
107    switch (Settings->typeOfObjectsSegmented) {
108        case Vision::Segment::Bright:
109            ui->rb_useDark_3->setChecked(false);
110            ui->rb_useLight_3->setChecked(true);
111            break;
112        case Vision::Segment::Dark:
113            ui->rb_useDark_3->setChecked(true);
114            ui->rb_useLight_3->setChecked(false);
115            break;
116        switch (Settings->morphFilterType) {
117            case Vision::MorphologicalFilter::CLOSE:
118                ui->rb_useClose_3->setChecked(true);
119                ui->rb_useDilate_3->setChecked(false);
120                ui->rb_useErode_3->setChecked(false);
121                ui->rb_useOpen_3->setChecked(false);
122                break;
123            case Vision::MorphologicalFilter::OPEN:
124                ui->rb_useClose_3->setChecked(false);
125                ui->rb_useDilate_3->setChecked(false);
126                ui->rb_useErode_3->setChecked(false);
127                ui->rb_useOpen_3->setChecked(true);
128                break;
129            case Vision::MorphologicalFilter::ERODE:
130                ui->rb_useClose_3->setChecked(false);
131                ui->rb_useDilate_3->setChecked(false);
132                ui->rb_useErode_3->setChecked(true);
133                ui->rb_useOpen_3->setChecked(false);
134                break;
135            case Vision::MorphologicalFilter::DILATE:
136                ui->rb_useClose_3->setChecked(false);
137                ui->rb_useDilate_3->setChecked(true);
138                ui->rb_useErode_3->setChecked(false);
139                ui->rb_useOpen_3->setChecked(false);
140                break;
141        }
142        ui->sb_adaptContrastFactor_3->setValue(Settings->
143            adaptContrastKernelFactor);
144        ui->sb_adaptContrastKernel_3->setValue(Settings->
145            adaptContrastKernelSize);
146        ui->sb_blurMask_3->setValue(Settings->blurKernelSize);
147        ui->sb_morphMask_3->setValue(Settings->filterMaskSize);
```

```

146     ui->sb_sigmaFactor_3->setValue(Settings->sigmaFactor);
147
148     // Setup the neural Network tab
149     NN = nn;
150     QPixmap NNpix("Images/feedforwardnetwork2.png");
151     ui->label_NNimage->setPixmap(NNpix);
152     ui->label_NNimage->setScaledContents(true);
153
154     ui->spinBox_InputNeurons->setValue(NN->GetInputNeurons());
155     ui->spinBox_HiddenNeurons->setValue(NN->GetHiddenNeurons()
156         );
157     ui->spinBox_OutputNeurons->setValue(NN->GetOutputNeurons()
158         );
159     ui->spinBox_Elitisme->setValue(NN->ElitismeUsedByGA);
160     ui->spinBox_MaxGen->setValue(NN->MaxGenUsedByGA);
161     ui->spinBox_PopSize->setValue(NN->PopulationSizeUsedByGA);
162     ui->doubleSpinBox_endError->setValue(NN->EndErrorUsedByGA)
163         ;
164     ui->doubleSpinBox_MutationRate->setValue(NN->
165         MutationrateUsedByGA);
166     ui->doubleSpinBox_Beta->setValue(NN->GetBeta());
167     ui->doubleSpinBox_maxWeight->setValue(NN->
168         MaxWeightUsedByGA);
169     ui->doubleSpinBox_MinWeight->setValue(NN->
170         MinWeightUsedByGA);
171     ui->checkBox_PredictShape->setChecked(Settings->
172         PredictTheShape);
173     ui->checkBox_revolt->setChecked(Settings->Revolution);
174
175     // Setup the preference tab
176     ui->lineEdit_NeuralNetFolder->setText(
177         QString::fromStdString(Settings->NNFolder));
178     ui->lineEdit_Printer->setText(
179         QString::fromStdString(Settings->StandardPrinter));
180     ui->lineEdit_Samplefolder->setText(
181         QString::fromStdString(Settings->SampleFolder));
182     ui->lineEdit_SendTo->setText(
183         (QString::fromStdString(Settings->StandardSentTo)));
184     ui->lineEdit_SettingFolder->setText(
185         QString::fromStdString(Settings->SettingsFolder));
186     ui->lineEdit_NeuralNet->setText(
187         QString::fromStdString(Settings->NNlocation));
188
189     if (openNN) {
190         ui->tabWidget->setcurrentIndex(3);
191     }
192     initfase = false;
193 }
194
195 DialogSettings::~DialogSettings() { delete ui; }
196
197 void DialogSettings::openTab(int newValue) {
198     if (newValue > ui->tabWidget->count()) {
199         ui->tabWidget->setcurrentIndex(newValue);
200     }
201 }

```

```
195
196 void DialogSettings::on_pushButton_RestoreDefault_clicked()
197 {
198     Settings->LoadSettings("Settings/Default.ini");
199 }
200
201 void DialogSettings::on_pushButton_Open_clicked() {
202     QString fn = QFileDialog::getOpenFileName(
203         this, tr("Open Settings"), QDir::homePath(), tr(
204             "Settings (*.ini)"));
205     if (!fn.isEmpty()) {
206         if (!fn.contains(tr(".ini")))
207             fn.append(tr(".ini"));
208     }
209     Settings->LoadSettings(fn.toStdString());
210 }
211
212 void DialogSettings::on_pushButton_Save_clicked() {
213     QString fn = QFileDialog::getSaveFileName(
214         this, tr("Save Settings"), QDir::homePath(), tr(
215             "Settings (*.ini)"));
216     if (!fn.isEmpty()) {
217         if (!fn.contains(tr(".ini")))
218             fn.append(tr(".ini"));
219     }
220 }
221
222 void DialogSettings::on_checkBox_Backlight_clicked(bool
223     checked) {
224     ui->tabWidget_Hardware->setTabEnabled(2, checked);
225     Settings->useBacklightProjection = checked;
226 }
227
228 void DialogSettings::
229     on_comboBox_Microscopes_currentIndexChanged(
230         const QString &arg1) {
231
232     if (!initfase) {
233         std::string selectedCam = arg1.toStdString();
234         Microscope->openCam(selectedCam);
235         Settings->defaultWebcam = selectedCam;
236
237         ui->comboBox_Resolution->clear();
238         QStringList Resolutions;
239         for (uint32_t i = 0; i < Microscope->SelectedCam->
240             Resolutions.size(); i++) {
241             Resolutions
242                 << Microscope->SelectedCam->Resolutions[i].
243                     to_string().c_str();
244         }
245         ui->comboBox_Resolution->addItems(Resolutions);
246         ui->comboBox_Resolution->setcurrentIndex(
247             Microscope->SelectedCam->SelectedResolution->ID);
248 }
```

```

244     }
245 }
246
247 void DialogSettings::
248     on_comboBox_Resolution_currentIndexChanged(int index) {
249     if (!initfase) {
250         Microscope->SelectedCam->SelectedResolution =
251             &Microscope->SelectedCam->Resolutions[index];
252         Settings->selectedResolution = index;
253     }
254 }
255 void DialogSettings::on_checkBox_useHDR_clicked(bool checked)
256 {
257     ui->spinBox_NoFrames->setDisabled(!checked);
258     ui->label_nf->setDisabled(!checked);
259     Settings->useHDR = checked;
260 }
261 void DialogSettings::SetCamControl(Hardware::Microscope::
262     Cam_t *selectedCam,
263                                         QSlider *Brightness,
264                                         QSlider *Contrast,
265                                         QSlider *Saturation,
266                                         QSlider *Hue,
267                                         QSlider *Sharpness) {
268     for (uint32_t i = 0; i < selectedCam->Controls.size(); i
269        ++) {
270         if (selectedCam->Controls[i].name.compare("Brightness")
271             == 0) {
272             Brightness->setMinimum(selectedCam->Controls[i].
273                 minimum);
274             Brightness->setMaximum(selectedCam->Controls[i].
275                 maximum);
276         } else if (selectedCam->Controls[i].name.compare(""
277             "Contrast") == 0) {
278             Contrast->setMinimum(selectedCam->Controls[i].minimum)
279                 ;
280             Contrast->setMaximum(selectedCam->Controls[i].maximum)
281                 ;
282         } else if (selectedCam->Controls[i].name.compare(""
283             "Saturation") == 0) {
284             Saturation->setMinimum(selectedCam->Controls[i].
285                 minimum);
286             Saturation->setMaximum(selectedCam->Controls[i].
287                 maximum);
288         } else if (selectedCam->Controls[i].name.compare(""
289             "Hue") == 0) {
290             Hue->setMinimum(selectedCam->Controls[i].minimum);
291             Hue->setMaximum(selectedCam->Controls[i].maximum);
292         } else if (selectedCam->Controls[i].name.compare(""
293             "Sharpness") == 0) {
294             Sharpness->setMinimum(selectedCam->Controls[i].minimum
295                 );
296             Sharpness->setMaximum(selectedCam->Controls[i].maximum
297                 );
298         }
299     }
300 }

```

```
281     }
282 }
283 }
284
285 void DialogSettings::on_spinBox_NoFrames_editingFinished() {
286     Settings->HDRframes = ui->spinBox_NoFrames->value();
287 }
288
289 void DialogSettings::
290     on_doubleSpinBox_LightLevel_editingFinished() {
291     Settings->lightLevel =
292         static_cast<float>(ui->doubleSpinBox_LightLevel->value
293         ());
294 }
295
296 void DialogSettings::on_checkBox_useRainbow_clicked(bool
297     checked) {
298     Settings->enableRainbow = checked;
299 }
300
301 void DialogSettings::on_checkBox_InvertEncoder_clicked(bool
302     checked) {
303     Settings->encInv = checked;
304 }
305
306 void DialogSettings::
307     on_horizontalSlider_BrightFront_valueChanged(int value) {
308     if (!initfase) {
309         Settings->Brightness_front = value;
310     }
311
312 void DialogSettings::
313     on_horizontalSlider_ContrastFront_valueChanged(int value)
314     {
315     if (!initfase) {
316         Settings->Contrast_front = value;
317     }
318 void DialogSettings::
319     on_horizontalSlider_SaturationFront_valueChanged(
320         int value) {
321     if (!initfase) {
322         Settings->Saturation_front = value;
323     }
324
325 void DialogSettings::
326     on_horizontalSlider_HueFront_valueChanged(int value) {
327     if (!initfase) {
```

```
327     Settings->Hue_front = value;
328 }
329 }
330 }
331 void DialogSettings::
332     on_horizontalSlider_SharpnessFront_valueChanged(
333     int value) {
334     if (!initfase) {
335         Settings->Sharpness_front = value;
336     }
337 }
338 void DialogSettings::
339     on_horizontalSlider_BrightProj_valueChanged(int value) {
340     if (!initfase) {
341         Settings->Brightness_proj = value;
342     }
343 }
344 void DialogSettings::
345     on_horizontalSlider_ContrastProj_valueChanged(int value)
346     {
347     if (!initfase) {
348         Settings->Contrast_proj = value;
349     }
350 }
351 void DialogSettings::
352     on_horizontalSlider_SaturationProj_valueChanged(
353     int value) {
354     if (!initfase) {
355         Settings->Saturation_proj = value;
356     }
357 }
358 void DialogSettings::
359     on_horizontalSlider_HueProj_valueChanged(int value) {
360     if (!initfase) {
361         Settings->Hue_proj = value;
362     }
363 }
364 void DialogSettings::
365     on_horizontalSlider_SharpnessProj_valueChanged(int value)
366     {
367     if (!initfase) {
368         Settings->Sharpness_proj = value;
369     }
370 }
371 void DialogSettings::on_cb_use_adaptContrast_3_clicked(bool
372 checked) {
373     Settings->useAdaptiveContrast = checked;
374     ui->sb_adaptContrastFactor_3->setDisabled(!checked);
375     ui->sb_adaptContrKernel_3->setDisabled(!checked);
376 }
```

```
374
375 void DialogSettings::on_cb_useBlur_3_clicked(bool checked) {
376     Settings->useBlur = checked;
377     ui->sb_blurMask_3->setDisabled(!checked);
378 }
379
380 void DialogSettings::on_rb_useDark_3_toggled(bool checked) {
381     if (checked) {
382         Settings->typeOfObjectsSegmented = Vision::Segment::Dark
383         ;
384     } else {
385         Settings->typeOfObjectsSegmented = Vision::Segment::
386             Bright;
387     }
388 }
389
390 void DialogSettings::on_cb_ignoreBorder_3_clicked(bool
391     checked) {
392     Settings->ignorePartialBorderParticles = checked;
393 }
394
395
396 void DialogSettings::on_sb_sigmaFactor_3_editingFinished() {
397     Settings->sigmaFactor = ui->sb_sigmaFactor_3->value();
398 }
399
400 void DialogSettings::on_rb_useOpen_3_clicked(bool checked) {
401     Settings->morphFilterType = Vision::MorphologicalFilter::
402         OPEN;
403 }
404
405 void DialogSettings::on_rb_useClose_3_clicked(bool checked)
406 {
407     Settings->morphFilterType = Vision::MorphologicalFilter::
408         CLOSE;
409 }
410
411
412 void DialogSettings::on_rb_useErode_3_clicked(bool checked)
413 {
414     Settings->morphFilterType = Vision::MorphologicalFilter::
415         ERODE;
416 }
417
418 void DialogSettings::on_sb_morphMask_3_editingFinished() {
419     Settings->filterMaskSize = ui->sb_morphMask_3->value();
420 }
```

```
419
420 void DialogSettings::on_spinBox_MaxGen_editingFinished() {
421     NN->MaxGenUsedByGA = ui->spinBox_MaxGen->value();
422 }
423
424 void DialogSettings::on_spinBox_PopSize_editingFinished() {
425     NN->PopulationSizeUsedByGA = ui->spinBox_PopSize->value();
426 }
427
428 void DialogSettings::
429     on_doubleSpinBox_MutationRate_editingFinished() {
430     NN->MutationrateUsedByGA = ui->doubleSpinBox_MutationRate
431         ->value();
432 }
433
434 void DialogSettings::
435     on_doubleSpinBox_EndError_editingFinished() {
436     NN->EndErrorUsedByGA = ui->doubleSpinBox_EndError->value()
437         ;
438 }
439
440 void DialogSettings::
441     on_doubleSpinBox_maxWeight_editingFinished() {
442     NN->MaxWeightUsedByGA = ui->doubleSpinBox_maxWeight->value()
443         ;
444 }
445
446 void DialogSettings::
447     on_doubleSpinBox_MinWeight_editingFinished() {
448     NN->MinWeightUsedByGA = ui->doubleSpinBox_MinWeight->value()
449         ;
450 }
451
452 void DialogSettings::on_doubleSpinBox_Beta_editingFinished()
453     {
454     NN->SetBeta(ui->doubleSpinBox_Beta->value());
455 }
456
457 void DialogSettings::on_spinBox_InputNeurons_editingFinished()
458     {
459     NN->SetInputNeurons(ui->spinBox_InputNeurons->value());
460 }
461
462 void DialogSettings::
463     on_spinBox_HiddenNeurons_editingFinished() {
464     NN->SetHiddenNeurons(ui->spinBox_HiddenNeurons->value());
465 }
466
467 void DialogSettings::
468     on_spinBox_OutputNeurons_editingFinished() {
469     NN->SetOutputNeurons(ui->spinBox_OutputNeurons->value());
470 }
```

```
463
464 void DialogSettings::on_pushButton_selectSampleFolder_clicked() {
465     QString fn = QFileDialog::getExistingDirectory(
466         this, tr("Select the Sample Directory"),
467         QString::fromStdString(Settings->SampleFolder),
468         QFileDialog::ShowDirsOnly | QFileDialog::DontResolveSymlinks);
469     if (!fn.isEmpty()) {
470         ui->lineEdit_Samplefolder->setText(fn);
471         Settings->SampleFolder = fn.toStdString();
472     }
473 }
474
475 void DialogSettings::on_pushButton_SelectSettingFolder_clicked() {
476     QString fn = QFileDialog::getExistingDirectory(
477         this, tr("Select the Setting Directory"),
478         QString::fromStdString(Settings->SettingsFolder),
479         QFileDialog::ShowDirsOnly | QFileDialog::DontResolveSymlinks);
480     if (!fn.isEmpty()) {
481         ui->lineEdit_SettingFolder->setText(fn);
482         Settings->SettingsFolder = fn.toStdString();
483     }
484 }
485
486 void DialogSettings::on_pushButton_SelectNNFolder_clicked()
487 {
488     QString fn = QFileDialog::getExistingDirectory(
489         this, tr("Select the NeuralNet Directory"),
490         QString::fromStdString(Settings->NNFolder),
491         QFileDialog::ShowDirsOnly | QFileDialog::DontResolveSymlinks);
492     if (!fn.isEmpty()) {
493         ui->lineEdit_NeuralNetFolder->setText(fn);
494         Settings->NNFolder = fn.toStdString();
495     }
496 }
497 void DialogSettings::on_pushButton_SelectNN_clicked() {
498     QString fn =
499         QFileDialog::getOpenFileName(this, tr("Select the
500                                     standard Neural Net"),
501                                     QDir::homePath(), tr(""
502                                     "NeuralNet (*.NN)"));
503     if (!fn.isEmpty()) {
504         if (!fn.contains(tr(".NN")))
505             fn.append(tr(".NN"));
506         Settings->NNlocation = fn.toStdString();
507         ui->lineEdit_NeuralNet->setText(fn);
508     }
509 }
510 void DialogSettings::on_spinBox_NoShots_editingFinished() {
```

```
511     Settings->StandardNumberOfShots = ui->spinBox_NoShots->
512         value();
513
514     void DialogSettings::on_checkBox_PredictShape_clicked(bool
515         checked) {
516         Settings->PredictTheShape = checked;
517     }
518     void DialogSettings::on_checkBox_revolt_clicked(bool checked
519     )
520     {
521         Settings->Revolution = checked;
522     }
```

N.0.30 Dialog Neural Network Class

```
1 #ifndef DIALOGNN_H
2 #define DIALOGNN_H
3
4 #include <QDialog>
5 #include "SoilMath.h"
6 #include "soilanalyzer.h"
7 #include "dialogsettings.h"
8 #include <qcustomplot.h>
9 #include <QDebug>
10
11 namespace Ui {
12     class DialogNN;
13 }
14
15 class DialogNN : public QDialog
16 {
17     Q_OBJECT
18
19 public:
20     explicit DialogNN(QWidget *parent = 0, SoilMath::NN *
21         neuralnet = nullptr, SoilAnalyzer::SoilSettings *
22         settings = nullptr, DialogSettings *settingWindow =
23         nullptr);
24     ~DialogNN();
25
26     private slots:
27     void on_pushButton_Settings_clicked();
28     void on_learnErrorUpdate(double newError);
29     void on_pushButton_SelectSamples_clicked();
30     void on_pushButton_Learn_clicked();
31     void on_pushButton_SaveNN_clicked();
32     void on_pushButton_OpenNN_clicked();
33     void on_actionAbort_triggered();
34
35     void setupErrorGraph();
36     void makeLearnVectors(InputLearnVector_t &input,
37         OutputLearnVector_t &output);
38
39     private:
40     Ui::DialogNN *ui;
41     DialogSettings *SettingsWindow = nullptr;
42     SoilMath::NN *NeuralNet = nullptr;
43     SoilAnalyzer::SoilSettings *Settings = nullptr;
44
45     QVector<double> currentError;
46     QVector<double> errorTicks;
47     double currentGeneration = 0;
```

```

51     QStringList fn;
52 };
53
54 #endif // DIALOGNN_H


---


1 #include "dialognn.h"
2 #include "ui_dialognn.h"
3
4 DialogNN::DialogNN(QWidget *parent, SoilMath::NN *neuralnet,
5                     SoilAnalyzer::SoilSettings *settings,
6                     DialogSettings *settingsWindow)
7     : QDialog(parent), ui(new Ui::DialogNN) {
8     ui->setupUi(this);
9
10    if (neuralnet == nullptr) {
11        neuralnet = new SoilMath::NN;
12    }
13    NeuralNet = neuralnet;
14    if (settings == nullptr) {
15        settings = new SoilAnalyzer::SoilSettings;
16    }
17    Settings = settings;
18    if (settingsWindow == nullptr) {
19        settingsWindow = new DialogSettings;
20    }
21    SettingsWindow = settingsWindow;
22
23    // Setup the Qplots
24    ui->widget_NNError->addGraph();
25    ui->widget_NNError->addGraph();
26
27    ui->widget_NNError->xAxis->setLabel("Generation [-]");
28    ui->widget_NNError->yAxis->setLabel("Error [%]");
29    QCPPlotTitle *widget_NNErrorTitle = new QCPPlotTitle(ui->
30        widget_NNError);
31    widget_NNErrorTitle->setText("Learning error");
32    widget_NNErrorTitle->setFont(QFont("sans", 10, QFont::Bold));
33    ui->widget_NNError->plotLayout()->insertRow(0);
34    ui->widget_NNError->plotLayout()->addElement(0, 0,
35        widget_NNErrorTitle);
36
37    // Connect the NN learn error
38    connect(NeuralNet, SIGNAL(learnErrorUpdate(double)), this,
39             SLOT(on_learnErrorUpdate(double)));
40 }
41
42 DialogNN::~DialogNN() { delete ui; }
43
44 void DialogNN::on_pushButton_Settings_clicked() {
45     SettingsWindow->openTab(2);
46     SettingsWindow->show();
47     setupErrorGraph();
48 }

```



```

98                         OutputLearnVector_t &output)
99                         {
100                         for (uint32_t i = 0; i < fn.size(); i++) {
101                             SoilAnalyzer::Sample sample;
102                             sample.Load(fn[i].toStdString());
103                             for_each(sample.ParticlePopulation.begin(), sample.
104                                 ParticlePopulation.end(),
105                                 [&](SoilAnalyzer::Particle &P) {
106                                     if (P.FFDescriptors.size() >= NeuralNet->
107                                         GetInputNeurons()) {
108                                         ComplexVect_t ffdesc;
109                                         for (uint32_t j = 0; j < NeuralNet->
110                                             GetInputNeurons(); j++) {
111                                             ffdesc.push_back(P.FFDescriptors[j]);
112                                         }
113                                         input.push_back(ffdesc);
114                                         Predict_t predict = P.Classification;
115                                         predict.OutputNeurons = SoilMath::
116                                             makeOutput(P.GetAngularity(), NeuralNet
117                                             ->GetOutputNeurons());
118                                         output.push_back(predict);
119                                     }
120                                 });
121                         }
122                     );
123                 }
124             );
125         }
126     }
127 }
128 }
129
130 void DialogNN::on_pushButton_SaveNN_clicked() {
131     QString fn = QFileDialog::getSaveFileName(
132         this, tr("Save NeuralNet"), QString::fromStdString(
133             Settings->NNFolder),
134             tr("NeuralNet (*.NN)"));
135     if (!fn.isEmpty()) {
136         if (!fn.contains(tr(".NN"))) {
137             fn.append(tr(".NN"));
138         }
139         NeuralNet->SaveState(fn.toStdString());
140     }
141 }
142
143 void DialogNN::on_pushButton_OpenNN_clicked() {
144     QString fn = QFileDialog::getOpenFileName(
145         this, tr("Open NeuralNet"),
146         QString::fromStdString(Settings->SampleFolder), tr(""
147             "NeuralNet (*.NN)"));
148     if (!fn.isEmpty()) {
149         if (!fn.contains(tr(".NN"))) {
150             fn.append(tr(".NN"));
151         }
152         if (NeuralNet != nullptr) {
153             delete NeuralNet;
154         }
155         NeuralNet->LoadState(fn.toStdString());
156         connect(NeuralNet, SIGNAL(learnErrorUpdate(double)),
157             this,
158             SLOT(on_learnErrorUpdate(double)));
159     }
160 }
```

```
145  }
146
147 void DialogNN::on_actionAbort_triggered()
148 {
149     NeuralNet->EndErrorUsedByGA = ui->widget_NNError->graph
150     (0)->data()->lastKey();
151 }
```
